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

ТРУДЫ НИИСИ РАН

ТОМ 10 № 5-6

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2020

Тематика номера:

Развитие суперкомпьютерных технологий и оптимизация суперкомпьютерных вычислений, проектирование и моделирование СБИС, моделирование физических процессов в микро- и наноэлектронике, моделирование динамических природных процессов, вопросы программирования, визуализа-ция, информационные технологии в учебной информатике

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:

Development of supercomputer technologies and optimization of supercomputer calculations, design and modeling of VLSI, modeling of physical processes in micro- and nanoelectronics, modeling of dy-namic natural processes, programming issues, visualization, information technology in educational in-formatics

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

Облачный сервис для высокопроизводительных вычислений на базе платформы OpenNebula

А.В. Баранов[1], Б.В. Долгов[2], Е.А. Киселёв[3], Д.С. Ляховец[4]

[1]Межведомственный суперкомпьютерный центр РАН, Москва, Россия, antbar@mail.ru;

[2]Межведомственный суперкомпьютерный центр РАН, Москва, Россия, borisdol@jscc.ru;

[3]Межведомственный суперкомпьютерный центр РАН, Москва, Россия, kiselev@jscc.ru;

[4]Межведомственный суперкомпьютерный центр РАН, Москва, Россия, anetto@inbox.ru

Аннотация. Статья посвящена разработке облачного сервиса для высокопроизводительных вычислений на базе платформы OpenNebula. Представлены архитектура и реализация облачного сервиса, отличающиеся от известных решений тем, что позволяют интегрировать в единое облако суперкомпьютерные ресурсы с сохранением традиционного режима коллективного пользования. Для этого традиционные системы управления заданиями суперкомпьютеров представляются в OpenNebula в виде виртуальных вычислительных узлов. Между узлами OpenNebula и суперкомпьютерными системами устанавливается взаимно однозначное соответствие. Распределение и запуск платформой OpenNebula виртуальной машины на такой вычислительный узел автоматически влечет постановку задания в очередь соответствующей узлу суперкомпьютерной системы. За счет организованной обратной связи платформа OpenNebula распределяет задания с учетом реальной загрузки суперкомпьютеров и осуществляет балансировку всей облачной инфраструктуры.

Ключевые слова: высокопроизводительные вычисления, облачные вычисления, суперкомпьютер, планирование заданий, облачный сервис, OpenNebula, СУППЗ

1. Введение

Сфера применения высокопроизводительных вычислений расширяется с каждым годом. Суперкомпьютерные технологии становятся одним из главных инструментов в передовых научных исследованиях и промышленных разработках. Число пользователей суперкомпьютеров неуклонно возрастает, что обуславливает актуальность представления типовых высокопроизводительных вычислений в виде научно-технического сервиса, доступного пользователям через Интернет из любой точки. Идеальным с точки зрения пользователя решением было бы представление услуг по высокопроизводительным вычислениям в виде облачных сервисов разных уровней (IaaS, PaaS, SaaS).

В работах [1-3] отмечены основные трудности, возникающие на пути построения облачных сервисов для высокопроизводительных вычислений. Эти трудности главным образом связаны с технологией гипервизорной виртуализации, применяемой при организации облачных вычислений. Гипервизорная виртуализация привносит существенные накладные расходы на вычисления [1, 2], что является неприемлемым для широкого класса задач, требующих высокопроизводительных расчетов. Кроме высоких накладных расходов, отмечаются [3] не всегда поддерживаемая гипервизорами уникальность суперкомпьютерных ресурсов, а также требование непосредственного доступа пользователя к суперкомпьютерным ресурсам для обеспечения максимальной эффективности использования вычислительных устройств и быстродействия суперкомпьютерной системы.

Вместе с указанными трудностями в работе [3] отмечено сходство характеристик суперкомпьютерных систем управления заданиями (СУЗ) и облачных вычислений. Для проведения высокопроизводительных расчетов пользователь оформляет т.н. вычислительное задание, которое представляет собой информационный объект, включающий расчетную программу, исходные данные и требования к ресурсам. Последние обеспечивают такие характерные для облачных вычислений свойства, как самообслуживание по требованию и эластичность – при направлении задания в СУЗ пользователь по своему усмотрению определяет объем требуемых ему ресурсов и время выполнения задания. Пользователь суперкомпьютера имеет к нему универсальный доступ по сети из любой точки и с любого оконечного устройства. СУЗ суперкомпьютера прозрачным для пользователя образом динамически выделяет вычислительные ресурсы для каждого задания, обеспечивая полноту и равномерную

загрузку суперкомпьютерной системы. При этом СУЗ ведет точный статистический учет, позволяющий выставлять счета только за фактически потребленные пользователями суперкомпьютерные ресурсы.

Фактически СУЗ суперкомпьютера выполняет серверные функции облачной платформы, обладая схожими характеристиками. Этот факт имеет несколько отрицательных последствий. Во-первых, требуется разработка клиентской части облачного сервиса. Во-вторых, СУЗ не распределяет вычислительную нагрузку между несколькими суперкомпьютерными системами. В-третьих, СУЗ несовместимы с облачными платформами, поскольку и те, и другие системы требуют полного контроля над управляемыми вычислительными ресурсами. В Межведомственном суперкомпьютерном центре РАН в последние годы ведутся активные разработки для преодоления указанных негативных моментов. Так, в работе [4] рассмотрен специально подобранный программный стек, на основе которого было создано веб-приложение, взаимодействующее с СУЗ нескольких суперкомпьютеров и обеспечивающее предоставление пользователю облачного SaaS-сервиса для высокопроизводительных вычислений. Предложенное решение получилось узкоспециализированным и несовместимым с известными облачными платформами, что фактически делало невозможным его развитие. В работе [3] был предложен метод совмещения двух потоков заданий: поступающих в СУЗ традиционным образом и поступающих через облачный сервис. Основу решения составила распространенная облачная платформа OpenStack [5, 6], при этом СУЗ представлялась для OpenStack в виде виртуального гипервизора, управляющего вычислительным узлом с большим числом ядер. Запуск заданий через облачный сервис осуществлялся OpenStack как запуск очередной виртуальной машины через специально разработанный драйвер СУЗ библиотеки libvirt [7]. Указанный драйвер [3] позволил совместить потоки заданий облачного сервиса и СУЗ, однако быстро выявилась проблема отсутствия обратной совместимости версий платформы OpenStack и библиотеки libvirt. Оказалось, что при выходе новой версии OpenStack или libvirt фактически необходимо разрабатывать новый драйвер СУЗ, что сводит на нет преимущества изложенного в [3] подхода.

В работе [8] рассмотрен подход к построению облачных сервисов видов PaaS и SaaS, основанных на совместном функционировании облачной платформы Proxmox VE [9] и системы управления прохождением параллельных заданий (СУППЗ) [10], применяемой в качестве СУЗ в МСЦ РАН. Подход применим для класса суперкомпьютерных приложений, нечувствительным к накладным расходам на виртуализацию.

Настоящая работа посвящена развитию исследований [3, 8] и рассматривает возможности построения облачных сервисов для высокопроизводительных вычислений на базе платформы OpenNebula [11].

2. Возможности платформы OpenNebula для организации высокопроизводительных вычислений

OpenNebula [11] представляет собой открытую и расширяемую платформу автоматизации работы центров обработки данных, на базе которой возможно построение собственной облачной инфраструктуры. Для этого OpenNebula осуществляет самостоятельное управление вычислительными и сетевыми ресурсами, ресурсами хранения данных, а также имеет возможность подключать к своей инфраструктуре ресурсы внешних облачных провайдеров. История OpenNebula ведет свое начало с исследовательского проекта 2005 года, осуществлявшегося И. Льоренте и Р. Монтеро. Как отмечено в [11], OpenNebula является результатом многолетних исследований и разработок, осуществлявшихся совместно с сообществом пользователей и участниками рынка облачных вычислений. Платформа обеспечивает эффективное управление виртуальными машинами и позволяет создавать масштабируемые облачные инфраструктуры. В состав платформы входят средства для развертывания виртуальных окружений, мониторинга, контроля доступа, обеспечения безопасности и управления хранилищем. Платформа свободно распространяется в открытых исходных кодах под лицензией Apache и может работать под управлением Linux известных версий: Ubuntu, openSUSE, RHEL/CentOS и Debian.

В мировой и отечественной исследовательской практике накоплен опыт применения платформы OpenNebula для организации высокопроизводительных вычислений. Так, в работе [12] платформа OpenNebula была применена для создания виртуальных высокопроизводительных вычислительных систем в целях обучения студентов. Возможности OpenNebula были использованы для упрощения процесса создания, развертывания и управления десятками и сотнями образов виртуальных машин, работающих на физическом высокопроизводительном кластере. Аналогичная работа была выполнена в ИММ

УрО РАН [13], где на базе OpenNebula была создана облачная инфраструктура для консолидации компьютерного оборудования, используемого в целях разработки, отладки и развертывания программного обеспечения, а также дидактической поддержки образовательных курсов.

Авторы программной системы Megh [14], созданной на платформе OpenNebula, провели тестирование своей системы по сценарию высокопроизводительных вычислений. Несколько экземпляров приложения было запущено на обычном сервере и в облачной среде, при этом результаты вычислений показали, что облачная платформа OpenNebula выполняет приложение с минимальными накладными расходами, демонстрируя хорошую производительность и минимальное время отклика.

Исследование [15] посвящено проблеме эластичного распределения ресурсов при высокопроизводительных вычислениях. В работе отмечается, что традиционные СУЗ суперкомпьютеров перекладывают задачу адаптации приложений к динамически изменяющейся вычислительной нагрузке на программистов, которые заранее должны включить механизмы балансировки нагрузки в исходные коды приложений. Работа [15] представляет AutoElastic – модель эластичности на уровне PaaS для высокопроизводительных вычислений в облаке. Предлагаемый подход заключается в обеспечении эластичности высокопроизводительных приложений без вмешательства пользователя или модификации исходного кода приложений. Макет AutoElastic основан на платформе OpenNebula, с помощью которой авторы [15] достигли 25% прироста производительности.

Следует отметить критические исследования, например, работу [16], в которой сравнивается производительность наиболее распространенных облачных платформ OpenStack, OpenNebula и Eucalyptus [17]. Результаты исследования [16] продемонстрировали существенное (по оценкам авторов, на порядок) отставание OpenNebula от конкурентов по времени запуска виртуализированной среды для высокопроизводительных вычислений. Применение иной методики тестирования может привести к другим результатам. Так, в работе [9] платформа OpenNebula показала наилучшую производительность среди всех рассмотренных свободно распространяемых решений.

OpenNebula с успехом применяется исследователями для организации высокопроизводительных вычислений в распределенных и грид-средах. В статье [18] рассмотрена грид-инфраструктура, построенная на базе облачного стека OpenNebula для эластичных приложений, которые автоматически масштабируются в соответствии с вычислительными потребностями. Как показывает исследование [19], OpenNebula может служить основой для федеративной интеграции нескольких облачных инфраструктур. Федеративное объединение строится по схеме «одна главная зона и несколько управляемых зон», где под «зоной» понимается каждая из инфраструктур федерации. В подобной интеграции все облачные инфраструктуры имеют единую базу пользователей, а управление всей федерацией осуществляется централизованно из главной зоны.

Наиболее ярким примером применения OpenNebula для организации высокопроизводительных вычислений может служить облачная инфраструктура Объединенного института ядерных исследований (ОИЯИ) [20, 21]. Для выполнения широкого спектра научных расчетов пользователям ОИЯИ предоставляется многофункциональный информационно-вычислительный комплекс (МИВК), включающий в себя различные вычислительные ресурсы ОИЯИ. Для ускорения выполнения научных расчетов, а также упрощения запуска однотипных задач, но с использованием различных типов ресурсов МИВК, на базе платформы OpenNebula в течение последних лет разрабатывается и совершенствуется облачный сервис ОИЯИ для научных и инженерных расчетов. Облачный сервис [21] реализует гибкую и модульную архитектуру, которая позволяет запускать большое количество задач на различных типах вычислительных архитектур.

Таким образом, мировой и отечественный опыт демонстрируют применимость платформы OpenNebula для организации высокопроизводительных вычислений, в т.ч. в распределенных средах. Новизна настоящей работы заключается в интеграции облачного сервиса на базе OpenNebula в исторически сложившуюся в МСЦ РАН систему коллективного пользования суперкомпьютерными ресурсами и реализации на базе OpenNebula метода совмещения потоков заданий, предложенного в статье [3].

3. Архитектура облачного сервиса для высокопроизводительных вычислений под управлением OpenNebula

Архитектура облачного сервиса для высокопроизводительных вычислений на базе платформы OpenNebula представлена на рисунке 1.

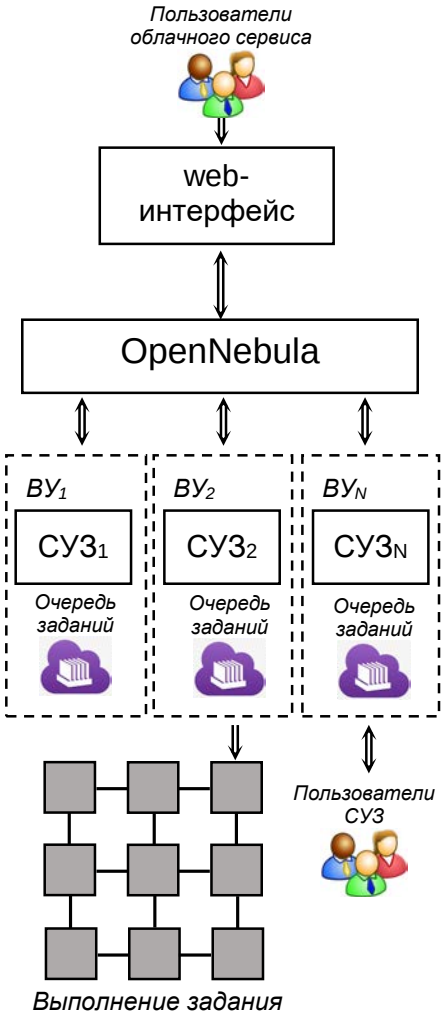


Рис. 1. Архитектура облачного сервиса для высокопроизводительных вычислений на базе платформы OpenNebula

Пользователи обращаются к облачному сервису традиционным образом через web-интерфейс, который может предлагать зарегистрированные сервисы вида SaaS или PaaS. Заполнив форму для производства расчетов и задав исходные данные, пользователь через web-интерфейс отправляет запрос на расчет платформе OpenNebula. Как видно из рисунка 1, под управлением OpenNebula находятся несколько систем управления заданиями (СУЗ 1, СУЗ 2, …, СУЗ N на рисунке 1) суперкомпьютера. Каждая СУЗ обслуживает свой собственный суперкомпьютер и ведет собственную (локальную) очередь заданий. Для OpenNebula каждая СУЗ представляется как единый виртуальный вычислительный узел, на котором с точки зрения OpenNebula возможен запуск виртуальных машин. С помощью специального разработанного авторами механизма запрос пользователя на производство высокопроизводительный расчетов преобразуется в OpenNebula в запрос на запуск виртуальной машины (либо контейнера).

Получив такой запрос, OpenNebula выбирает наименее загруженный виртуальный узел и запускает на нем виртуальную машину (контейнер). Далее этот запуск преобразуется в обращение к СУЗ, соответствующей выбранному OpenNebula виртуальному узлу. По этому обращению происходит формирование и постановка в очередь СУЗ суперкомпьютерного задания, которое будет реализовывать (выполнять) запрос пользователя, пришедший от облачного сервиса. После прохождения очереди СУЗ задание запускается на ресурсах суперкомпьютерной системы, производит необходимые расчеты и завершается. Результаты возвращаются пользователю через web-интерфейс.

Особенность предложенной архитектуры состоит в том, что одновременно с OpenNebula с каждой СУЗ продолжают работать локальные пользователи: они ставят свои задания в локальную очередь СУЗ, эти задания выполняются наряду с заданиями облачного сервиса, пользователи получают результаты обычным для той или иной СУЗ образом. Очевидно, локальные пользователи загружают суперкомпьютерную систему, и для обеспечения адекватного выбора платформой OpenNebula наименее загруженного виртуального узла необходима обратная связь между СУЗ и OpenNebula. В этом случае рост или снижение загрузки суперкомпьютерной системы будет представляться в OpenNebula как рост или снижение загрузки соответствующего суперкомпьютеру виртуального узла OpenNebula.

Таким образом, предложенная архитектура позволяет совместить по управлению поток заданий от облачного сервиса и локальный поток заданий от СУЗ суперкомпьютера. В этом рассматриваемая архитектура соответствует предложенному в [3] методу, реализованному на базе платформы OpenStack. Однако, при реализации архитектуры выяснился ряд аспектов, не позволивших напрямую использовать метод [3] и потребовавших от авторов отдельных технических решений.

4. Реализация облачного сервиса под управлением OpenNebula на ресурсах МСЦ РАН

Рассмотренный в [3] метод предполагал возможность представления СУЗ в виде виртуального узла путем разработки соответствующего драйвера библиотеки libvirt, к которой с запросами обращается облачная платформа. Данное

предположение было справедливо для использовавшейся в [3] платформы OpenStack. OpenNebula тоже обращается к функциям libvirt, только в отличие от OpenStack, это обращение осуществляется непосредственно на вычислительном узле. В такой ситуации с учетом отсутствия обратной совместимости версий libvirt разработка драйвера СУЗ аналогично работе [3] теряет смысл.

Вместо разработки драйвера libvirt был предложен следующий подход. Как показано на рисунке 2, под управление OpenNebula выделяются виртуальные вычислительные узлы (на рисунке 2 показан i-й виртуальный вычислительный узел $ВУ_i$), каждый узел соответствует одной СУЗ, в качестве которой при реализации макета применялась эксплуатируемая в МСЦ РАН СУППЗ. Обращение пользователя к облачному сервису осуществляется через web-интерфейс, который подготавливает и сохраняет исходные данные и требования пользователя в специальной базе данных (БД) под управлением выделенного сервера БД. После подготовки данных следует запрос к OpenNebula, получив который, OpenNebula выбирает наиболее свободный виртуальный узел и запускает на нем виртуальную машину из образа, соответствующего выбранному пользователем облачному сервису.

Образ виртуальной машины облачного сервиса представляет собой минимальный набор из ОС Linux и утилит, достаточный для выполнения командного файла – сценария запуска задания. В развернутой на узле виртуальной машине (на рисунке 2 стартовавшие виртуальные машины обозначены Job_1, Job_2, Job_3, Job_4) начинает выполнение командный сценарий запуска задания. Этот сценарий формирует вычислительное задание для СУППЗ, используя исходные данные, которые были сохранены web-интерфейсом в БД, после чего обращается к соответствующей виртуальному узлу СУППЗ (на рисунке 2 виртуальному узлу $ВУ_i$ соответствует $СУППЗ_i$) на постановку сформированного задания в очередь.

После прохождения через очередь СУППЗ задание выполняется на суперкомпьютерной системе. Результаты выполнения сохраняются в базе данных, через которую становятся доступными пользователю в web-интерфейсе.

Для эффективной работы рассмотренной схемы реализации необходимо решить следующие задачи.

1. Определить критерий, на основе которого будет осуществляться выбор суперкомпьютера и соответствующей ему СУППЗ для распределения сформированного задания. Критерий должен оперировать некоторым коэффициентом загрузки, формируемым на основе текущих характеристик СУППЗ (таких как количество заданий в очереди, время освобождения необходимых ресурсов и т.п.).

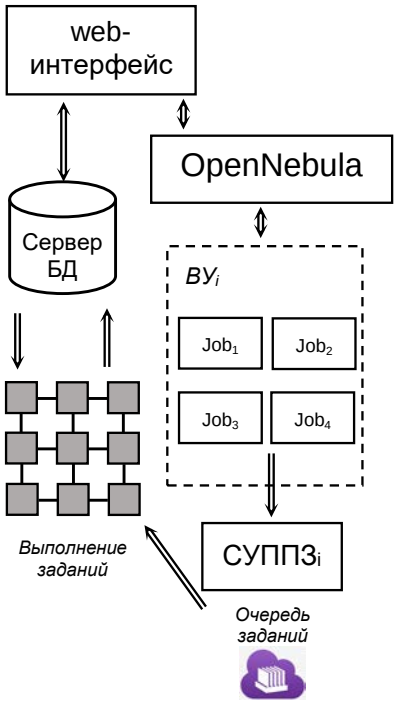


Рис. 2. Макет облачного сервиса для высокопроизводительных вычислений на базе платформы OpenNebula

2. Настроить каждый виртуальный вычислительный узел OpenNebula таким образом, чтобы он выполнял запрос к связанной с ним СУППЗ, получал ее характеристики, формировал на основе этих характеристик коэффициент загрузки и отправлял сформированный коэффициент на управляющий узел OpenNebula

3. Настроить управляющий узел OpenNebula таким образом, чтобы он принимал от вычислительных узлов OpenNebula коэффициенты загрузки и с помощью критерия принимал решение о распределении очередного задания пользователя на тот или иной вычислительный узел OpenNebula, т.е. в ту или иную суперкомпьютерную систему под управлением СУППЗ.

4. Создать подсистему запуска задания, которая при распределении задания на вычислительный узел OpenNebula добавила бы задание в очередь СУППЗ.

5. Создать подсистему настройки параметров пользовательского задания, позволяющую определять требования к ресурсам: количество узлов суперкомпьютера, число процессорных ядер и пр.

Согласно документации OpenNebula, каж-

дому вычислительному узлу при его инициализации присваивается драйвер мониторинга. Драйвер мониторинга представляет собой набор командных сценариев, которые копируются с управляющего узла OpenNebula. Эти сценарии запускаются по заранее установленному расписанию, собирают данные о системе и передают их на управляющий узел OpenNebula. В документации описан процесс, позволяющий разработать собственный драйвер мониторинга и добавить дополнительные метрики, которые OpenNebula будет получать с вычислительных узлов. Было принято решение создать собственный драйвер мониторинга "SUPPZ_MONIT_DRV". Этот драйвер, помимо стандартных сценариев мониторинга, имеет в составе дополнительный сценарий, который через протокол ssh соединяется с СУППЗ, получает характеристики для вычисления коэффициента загрузки суперкомпьютера, вычисляет и отправляет коэффициент загрузки на управляющий узел OpenNebula.

Как показал практический опыт авторов, рассмотренный механизм не работает так, как указано в документации. Оказалось, что вычисленный коэффициент загрузки суперкомпьютера невозможно использовать на управляющем узле OpenNebula для принятия решения о распределении пользовательского задания. Исследования подтвердили, что коэффициент загрузки верно передается по tcp-соединению управляющему узлу OpenNebula, отображается в журналах, но при вычислении рейтинга виртуальных вычислительных узлов для распределения задания принимается равным нулю.

В результате было принято решение вместо создания новой метрики (коэффициента загрузки) переопределить значение одной из стандартных метрик OpenNebula и записывать туда вычисленное значение коэффициента загрузки суперкомпьютера. Так как задание пользователя не запускается непосредственно на вычислительном узле OpenNebula, и большой нагрузки на сеть не возникает, было принято решение использовать стандартную метрику "NETRX" (количество принятых по сети байт). Данное решение работает, но может легко ввести в заблуждение администраторов OpenNebula, которые не знакомы с проблемой создания собственной метрики мониторинга.

Решение о распределении задания на тот или иной вычислительный узел OpenNebula принимает планировщик OpenNebula (OpenNebula sched). В конфигурационном файле OpenNebula sched необходимо указать, что будет использоваться собственная политика для распределения заданий. Это делается путем задания параметра POLICY равным 3. Для собственной политики распределения заданий также необходимо указать параметр RANK, который будет вычисляться для каждого узла, и на его основе (меньшее значение приоритетнее) будет выбран виртуальный вычислительный узел для распределения на него пользовательского задания. Для разработанного макета облачного сервиса параметр RANK равен метрике NETRX, которая, в свою очередь, хранит значение коэффициента загрузки каждого суперкомпьютера под управлением СУППЗ. Каждый раз, когда пользователь запускает задание, планировщик OpenNebula рассчитывает рейтинг каждого вычислительного узла (представляющего отображение соответствующей СУППЗ) и распределяет задание на наименее загруженный суперкомпьютер согласно коэффициенту загрузки.

5. Заключение

Прямая реализация ранее предложенного метода совмещения потоков заданий от облачного сервиса от локальной СУЗ суперкомпьютера [3] оказалась затрудненной по причине отсутствия обратной совместимости версий облачной платформы OpenStack и библиотеки libvirt, что потребовало повторной реализации драйвера СУЗ для каждой новой версии OpenStack и libvirt. В качестве варианта решения в статье предложены архитектура и реализация облачного сервиса для высокопроизводительных вычислений на базе платформы OpenNebula. Суперкомпьютерные системы и их СУЗ представляются для платформы OpenNebula в виде витртуальных вычислительных узлов. Между вычислительными узлами OpenNebula и СУЗ суперкомпьютеров устанавливается биективное отображение. Обращение пользователя к облачному сервису влечет за собой запуск на вычислительном узле OpenNebula служебной виртуальной машины, внутри которой разворачивается сценарий формирования суперкомпьютерного задания и направления его в очередь соответствующей узлу СУЗ. При этом на виртуальном вычислительном узле функционирует разработанный авторами драйвер мониторинга, преобразующий информацию о реальной загрузке суперкомпьютера в информацию о загрузке и доступности вычислительного узла OpenNebula. На основе этой информации платформа OpenNebula распределяет облачные задания между подконтрольными СУЗ, автоматически балансируя загрузку суперкомпьютеров.

Публикация подготовлена в рамках работ по проектам РФФИ (гранты №№ 18-07-01325 и 19-07-01088).

Cloud HPC Service Based on OpenNebula

A.V. Baranov, B.V. Dolgov, E.A. Kiselev, D.S. Lyakhovets

Abstract. The article is devoted to the development of a high performance computing cloud service based on the OpenNebula platform. Presented cloud HPC service architecture and implementation allow integration of supercomputer resources into a single cloud while maintaining the traditional job queue mode. To do this, the traditional supercomputer job management systems are mapped into OpenNebula as virtual computational nodes. A one-to-one correspondence is established between the OpenNebula nodes and supercomputer systems. Virtual machines distribution and launch by OpenNebula on such a computational node entails job submitting in the corresponding supercomputer queue. Using feedback, the OpenNebula platform distributes jobs taking into account the supercomputer real load and balances the entire cloud infrastructure.

Keywords: HPC, cloud computing, supercomputer, job scheduling, cloud service, OpenNebula

Литература

1. А.О. Кудрявцев, В.К. Кошелев, А.О. Избышев, И.А. Дудина, Ш.Ф. Курмангалеев, А.И. Аветисян, В.П. Иванников, В.Е. Велихов, Е.А. Рябинкин. Разработка и реализация облачной системы для решения высокопроизводительных задач. «Труды Института системного программирования РАН», Т. 24 (2013), 13–34.

2. O. Aladyshev, A. Baranov, R. Ionin, E. Kiselev, B. Shabanov. Variants of deployment the high performance computing in clouds. “2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)”, Russia, Moscow, 2018, 1453–1457, https://doi.org/10.1109/EIConRus.2018.8317371

3. О.С. Аладышев, А.В. Баранов, А.П. Овсянников, Г.А. Балаян, В.С. Синицин. Методы и средства совмещения потоков заданий от облачных платформ и менеджеров управления ресурсами суперкомпьютера. «Программные продукты, системы и алгоритмы», (2018), № 4, 54–63, https://doi.org/10.15827/2311-6749.29.337

4. А.В. Баранов, А.А. Зонов. Вариант организации облачного сервиса для высокопроизводительных вычислений. «Программные системы: теория и приложения», (2016), №7:3(30), 3–23, https://doi.org/10.25209/2079-3316-2016-7-3-3-23

5. T. Rosado, J. Bernardino. An overview of openstack architecture. “18th International Database Engineering & Applications Symposium (IDEAS '14)”. USA, New York, 2014, 366–367, https://doi.org/10.1145/2628194.2628195.

6. M. Bist, M. Wariya, A. Agarwal. Comparing delta, open stack and Xen Cloud Platforms: A survey on open source IaaS. “2013 3rd IEEE International Advance Computing Conference (IACC)”, Ghaziabad, 2013, 96–100, https://doi.org/10.1109/IAdCC.2013.6514201.

7. W. Ashley. Foundations of Libvirt Development. Apress, Berkeley, CA, 2019, https://doi.org/10.1007/978-1-4842-4862-1.

8. Е.А. Киселев, А.В. Баранов. Облачные сервисы для научных высокопроизводительных вычислений на базе платформы Proxmox. «Вычислительные технологии», Т. 24 (2019), № 6, 5–12, https://doi.org/10.25743/ICT.2019.24.6.002.

9. G. Rycaj. Comparison of virtualization performance of Proxmox, OpenVZ, OpenNebula, Vmware ESX and Xen Server. «Journal of Computer Sciences Institute», V. 12 (2019), 214–219, https://doi.org/10.35784/jcsi.490.

10. G. Savin, B. Shabanov, P. Telegin, A. Baranov. Joint Supercomputer Center of the Russian Academy of Sciences: Present and Future. “Lobachevskii Journal of Mathematics”, V. 40 (2019), № 11, 1853–1862, https://doi.org/10.1134/S1995080219110271.

11. OpenNebula. Национальная библиотека им. Н.Э. Баумана, 2017, https://ru.bmstu.wiki/OpenNebula.

12. J. St. John, T. Hacker. Developing Virtual Clusters for High Performance Computing Using OpenNebula. “2012 ASEE Annual Conference & Exposition”, USA, San Antonio, 2012, https://doi.org/10.18260/1-2--21197.

13. Д.Г. Ермаков, Д.А. Усталов. Экспериментальная среда облачных вычислений в институте математики и механики УрО РАН. «Программные продукты и системы», (2012), №4, 110–115.

14. T. Bhardwaj, M. Kumar, S.C. Sharma. Megh: A Private Cloud Provisioning Various IaaS and SaaS.

“Soft Computing: Theories and Applications. Advances in Intelligent Systems and Computing”, V. 584 (2018), 485–494, https://doi.org/10.1007/978-981-10-5699-4_45.

15. R. Righi, V. Rodrigues, C. da Costa, G. Galante, L. de Bona, T. Ferreto. AutoElastic: Automatic Resource Elasticity for High Performance Applications in the Cloud. “IEEE Transactions on Cloud Computing”, 2016, vol. 4, no. 1, 6–19, https://doi.org/10.1109/TCC.2015.2424876.

16. M. Jones et al. Scalability of VM provisioning systems. “2016 IEEE High Performance Extreme Computing Conference (HPEC)”, USA, Waltham, 2016, 1–5, https://doi.org/10.1109/HPEC.2016.7761629.

17. D. Milojicic, R. Wolski. Eucalyptus: Delivering a Private Cloud. “Computer”, V. 44 (2011), N. 4, 102–104, https://doi.org/10.1109/MC.2011.109.

18. S. Bagnasco, D. Berzano, S. Lusso, M. Masera, S. Vallero. Managing competing elastic Grid and Cloud scientific computing applications using OpenNebula. «Journal of Physics: Conference Series», V. 664 (2015), https://doi.org/10.1088/1742-6596/664/2/022004.

19. А.В. Баранов, В.В. Кореньков, В.В. Юрченко, Н.А. Балашов, Н.А. Кутовский, Р.Н. Семёнов, С.Я. Свистунов. Подходы к интеграции облачных инфраструктур. «Компьютерные исследования и моделирование», Т. 8 (2016), № 3, 583–590.

20. Н.А. Балашов, А.В. Баранов, В.В. Кореньков, Н.А. Кутовский, А.В. Нечаевский, Р.Н. Семенов. Облачный сервис ОИЯИ: статус и перспективы. «Труды Института системного программирования РАН», Т. 27 (2015), № 6, 345–354, https://doi.org/10.15514/ISPRAS-2015-27(6)-22.

21. И.А. Соколов, В.В. Кореньков. Облачный сервис ОИЯИ для научных и инженерных расчетов на ресурсах МИВК. «Системный анализ в науке и образовании», (2019), № 2, 63–73.

Некоторые вопросы федеративной аутентификации в распределенной сети суперкомпьютерных центров

Ал.А. Гончар[1], Ю.Н. Морин[2], А.П. Овсянников[3]

[1]МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия, alex.gonchar@jscc.ru;

[2]МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия, morin@jscc.ru;

[3]МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия, ovsyannikov@jscc.ru

Аннотация. В статье рассматривается вопросы построения на основе федеративного подхода системы аутентификации, авторизации и учёта в распределенной сети суперкомпьютерных центров. Рассмотрены сценарии обработки вычислительных заданий в сети суперкомпьютерных центров, определён состав данных, предоставляемых организацией — клиентом суперкомпьютерному центру при обработке вычислительных заданий и передаваемых при перераспределении задания внутри сети суперкомпьютерных центров. Предложена схема распределенной авторизации в сети суперкомпьютерных центров на основе развёрнутой в Национальной исследовательской компьютерной сети удостоверяющей федерации RUNNetAAI и распределенной службы управления проектами организаций — клиентов сети СКЦ. Рассмотренный подход значительно упрощает применение федеративной системы аутентификации и авторизации для организаций — клиентов распределенной сети СКЦ.

Ключевые слова: распределенная сеть суперкомпьютерных центров, федеративная аутентификация, авторизация и учёт, удостоверяющая федерация, RUNNetAAI

1. Введение

Развитие и поддержка функционирования центров коллективного пользования научно-технологическим оборудованием, к которым относятся научные суперкомпьютерные центры предусматривается стратегией научно-технологического развития Российской Федерации и национальным проектом «Наука». Одной из устойчивых тенденций развития научных суперкомпьютерных центров (СКЦ) в Российской Федерации [1,2] является их объединение в территориально распределенную систему для повышения эффективности использования вычислительных ресурсов. Это, в частности, позволит обеспечить увеличения значений индикаторных показателей, предусмотренных Постановлением Правительства Российской Федерации от 17.05.2016 года №429 (http://government.ru/docs/23110/).

Работы над проектом распределенной сети научных суперкомпьютерных центров ведутся в Межведомственном суперкомпьютерном центре РАН [3]. Проект включает исследования и разработку методов планирования в глобальной очереди пользовательских заданий и перераспределения вычислительной нагрузки [1,4-7], создания децентрализованной системы управления заданиями и ресурсами [8-10], организации распределенных систем хранения данных [11,12], мониторинга состояния и загруженности ресурсов СКЦ [13], единой системы аутентификации [14,15], организации облачных сервисов для высокопроизводительных вычислений [16-25], объединения суперкомпьютерных центров на базе научной телекоммуникационной сети [2,26,27].

В [14] было показано, что единая система аутентификации распределенной сети СКЦ может быть построена на основе методов федеративной аутентификации. Отметим, что требование интеграции с системами федеративной аутентификации научных организаций присутствует в п. C.3.2.4 Спецификации требований к инфраструктуре XSEDE (США) [28], а в рамках проекта AARC [29] разрабатываются проекты федеративной аутентификации для европейской инфраструктуры суперкомпьютерных приложений PRACE [30]. Это подтверждает актуальность и перспективность предложенного в [14] подхода.

Однако федеративную аутентификацию следует рассматривать лишь как один из аспектов организации единой системы доступа. В настоящей работе рассматривается схема и методы организации такой системы, как комплексной системы аутентификации, авторизации и учёта, во взаимодействии с децентрализованной системой управления заданиями и ресурсами распределенной сети СКЦ, в контексте возможных сцена-

риев организации работы пользователя и обработки заданий.

2. Организация обработки заданий в распределенной сети СКЦ

В распределенной сети суперкомпьютерных центров коллективного пользования участвуют объединённые научной телекоммуникационной сетью центры, принадлежащие разным организациям и управляемые независимо [3, 2].

Для осуществления расчётов в СКЦ пользователь должен оформить так называемое вычислительное задание, включающее в себя прикладную расчётную программу, требования к ресурсам и исходные данные [9]. В каждый из СКЦ поступает свой поток пользовательских заданий, часть из которых может быть выполнена непосредственно на ресурсах принявшего задание СКЦ, а другая часть – перенаправлена в глобальную очередь, доступную всем СКЦ сети.

Разрабатываемая в рамках проекта [3] децентрализованная автоматизированная система управления заданиями и ресурсами в распределённой сети СКЦ должна обеспечить оперативное перераспределение вычислительной нагрузки в сети СКЦ. Согласно [3, 4, 9] в системе выделяется два уровня управления – нижний - локальный и верхний - глобальный. На локальном уровне осуществляется распределение заданий в рамках одной отдельной суперкомпьютерной установки или её раздела с отдельной локальной очередью заданий. Локальный уровень представлен системами управления заданиями (СУЗ) отдельной суперкомпьютерной установки, такими как SLURM [31], PBS [32], Moab [33] или СУППЗ [34]. На глобальном уровне посредством глобальной очереди заданий осуществляется распределение заданий между суперкомпьютерными установками, входящими в состав сети СКЦ. Методы и алгоритмы распределения заданий глобальной очереди, научно-технические аспекты реализации глобальной системы управления представлены в [6, 7, 1] и являются одним из направлений исследований проекта [9].

Единицей управления в составе распределённой сети СКЦ, является вычислительная установка – суперкомпьютер или его раздел, управляемый локальной СУЗ.

В состав каждой вычислительной установки входят:

– сервер доступа;

– управляющий сервер;

– решающее поле.

Решающее поле включает в себя несколько вычислительных модулей (ВМ), объединённых высокоскоростной коммуникационной сетью.

На сервере доступа и ВМ смонтированы файловые системы локальных (в рамках СКЦ) или распределенных СХД для хранения программ и исходных данных, а также для записи результатов расчётов.

Для подготовки и запуска задания пользователь авторизуется на сервере доступа и, используя его командный интерфейс, оформляет вычислительное задание и ставит его в очередь на исполнение. Как правило, для этого используются вызовы программ локальной СУЗ в командной строке сервера доступа, однако рассматриваются и варианты облачного сервиса, в том числе с использованием веб-интерфейса для запуска заданий [16-19].

Отметим, что процесс выполнения высокопроизводительных вычислений может включать этапы подготовки исходных данных (предобработка), собственно расчётов на суперкомпьютерах, визуализации и анализа результатов (постобработка). На каждом из этапов может использоваться наиболее подходящий вид вычислительной техники, так что этапы могут выполняться в разных СКЦ.

Обработка заданий в сети СКЦ требует обеспечения авторизации пользователя на сервере доступа, контроля прав доступа пользователя к файлам программ и данных, контроля использования вычислительных ресурсов (машинного времени). Входящие в сеть СКЦ управляются независимо и принадлежат разным организациям, соответственно пользователи имеют в разных СКЦ разные учётные записи, каждый суперкомпьютерный центр ведёт свою собственную базу (реестр) пользователей, отдельный учёт по потреблённым суперкомпьютерным ресурсам. Кроме того, одна организация - клиент СКЦ - может вести научные исследования по разным темам (проектам), использовать в одном проекте одновременно ресурсы нескольких СКЦ. Наконец, один пользователь может работать одновременно по нескольким темам, соответственно расход им суперкомпьютерных ресурсов должен учитываться по-разному.

Будем считать [14], что взаимоотношения организации-клиента и СКЦ строятся на основе договоров, учитывающих потребление суперкомпьютерных ресурсов по научным проектам (темам исследований). В рамках одного проекта может быть заключено несколько договоров на использование суперкомпьютерных ресурсов (например, в разное время), с раздельным их учёт по договорам, однако с теми же пользователями, использующими те же программы, файлы и каталоги.

С точки зрения авторизации и учёта в рамках СКЦ проект можно определить, как сущность, характеризующуюся:

- совокупностью проектных каталогов на доступных в данном СКЦ файловых системах (локальных или распределенных);

- доступной квотой пространства для хранения данных проекта; на доступных файловых системах;

- сроком (датой окончания) хранения данных проекта на доступных файловых системах;

- совокупностью учётных записей пользователей, работающих в рамках проектов;

- квотами по видам вычислительных ресурсов, которые можно использовать в рамках проекта (набором квот при раздельном учёте по разным договорам);

- набором данных, характеризующих объём и вид, использованных проекта ресурсов (в том числе с раздельным учётом).

Вычислительное задание на суперкомпьютерном ресурсе запускается от имени пользователя, поэтому для успешного запуска и выполнения задания необходимо:

1) идентифицировать пользователя;

2) связать задание пользователя с проектом (а в случае раздельного учёта ресурсов по договорам внутри проекта, ещё и с договором);

3) связать пользователя с учётной записью для работы с запрошенным ресурсом (проверить наличие учётной записи; возможно, создать временную учётную запись, если это предусмотрено регламентами использования ресурсов);

4) связать местонахождение исходных данных и программы с проектными каталогами, наличие необходимых прав доступа для чтения данных и записи результатов, а возможно обеспечить загрузку исходных данных в нужные проектные каталоги, выгрузку из них результатов и освобождение пространства на файловой системе;

5) определить, может ли задание выполняться на запрошенном вычислительном ресурсе: имеются ли у пользователя права на запуск задания в рамках данного проекта, имеется ли у проекта и пользователя достаточная квота использования вычислительных ресурсов для его выполнения;

6) учесть результаты выполнения вычислительного задания в части расходования вычислительных ресурсов и ресурсов систем хранения данных, обновить соответствующие квоты.

Сценарий обработки вычислительного задания зависит от следующих факторов:

места формирования (подготовки и запуска) задания: на сервере доступа внутри данного СКЦ; на общем сервере доступа сети СКЦ;

связи пользователя с проектом и договором: указана явно, например, при запуске задания или авторизации на сервере доступа; указана неявно, например, привязкой к проектному каталогу;

места нахождения исходных данных и месту выгрузки результатов: в локальном СКЦ (где формируется задание); в другом СКЦ; в распределенной файловой системе, доступной из всех СКЦ; в организации — клиенте;

перераспределения задания в другой СКЦ;

регламента использования учётных записей пользователей: необходимо наличие постоянной учётной записи пользователя для запуска на вычислительном ресурсе; учётная запись может быть создана «по требованию» и будет в будущих запусках связываться с данным пользователем; создаётся временная учётная запись и удаляется после завершения задания и выгрузки его результатов.

Ниже приведён пример реализованного в пилотном проекте СКЦ сценария (задание запускается через локальную систему управления заданиями, связь с проектом — неявная, место нахождения исходных данных — локальный СКЦ, задание перераспределяется из локального в «удалённый» СКЦ).

1) Пользователь идентифицируется при авторизации на сервере доступа вычислительного ресурса СКЦ.

2) Принадлежность задания проекту определяется проектным каталогом, в котором запускается (формируется) задание.

3) Учётные записи пользователей в разных СКЦ либо синхронизированы, либо используется глобальная таблица, в которой определено соответствие учётных записей одного пользователя в разных СКЦ. Перераспределенное через глобальную очередь задание может быть запущено только в СКЦ, в котором имеется требуемая учётная запись.

4) Исходные данные копируются из локального СКЦ на ресурс, в котором планируется выполнение задания, паспорт задания корректируется соответствующим образом; после выполнения задания результаты копируются из «удалённого» СКЦ в локальный по первоначальному месту записи результатов.

5) Права доступа к вычислительному ресурсу на «удалённом» СКЦ определяется наличием локальной учётной записи пользователя на ресурсе и квотами использования вычислительных ресурсов в рамках проекта, что проверяется локальной системой очередей «удалённого» СКЦ. Возможность (готовность) выполнения задания «удалённым» СКЦ сообщается его локальной системой управлением заданиями глобальной системе управления заданиями.

6) Расходование вычислительных ресурсов заданием по его завершению учитывается системой управления заданиями «удалённого» СКЦ, которая информирует об этом глобальную си-

стему управления заданиями, а та в свою очередь систему управления заданиями локального СКЦ, на котором формировалось задание.

3. Организация обработки заданий в распределенной сети СКЦ

Как отмечено в [14] ведение учётных записей пользователей непосредственно в СКЦ сопряжено с проблемой получения и обработки персональных данных пользователя. В соответствии с Федеральным законом «О персональных данных» от 27.07.2006 N 152-ФЗ СКЦ должен обеспечить защиту персональных данных, что для научного центра является непрофильной задачей. В то же время квотирование и учёт использованных суперкомпьютерных ресурсов в СКЦ ведётся в отношении организации — клиента СКЦ, эти же действия в отношении отдельного пользователя — суть внутренний вопрос управления проектом, который должен решаться организацией - клиентом.

Формирование агрегированной статистики, запрашиваемой учредителем (количество пользователей до 39 лет, доля пользователей с учёной степенью и т. д.) может быть выполнено на основе обезличенных данных, получаемых от организации — клиента. При необходимости срочного взаимодействия с пользователем в связи с обработкой его задания или компьютерным инцидентом, это взаимодействие может быть организовано через организацию — клиента по уполномоченным ею контактам.

В [14] предложен способ использования федеративной аутентификации для делегирования идентификации пользователя организации — клиенту (см. рис. 1).

Рис. 1. Авторизация в сети СКЦ на основе серверов аутентификации и учёта проекта в организации - клиенте

При обращении (1) к Серверу авторизации СКЦ пользователь, точнее используемая им программа (User Agent - UA), например, браузер, передаёт свой уникальный идентификатор - eduPersonPrincipalName (ePPN) [35] и публичный ключ pkey, который будет использоваться для доступа к серверу доступа вычислительной установки по протоколу ssh. На основе ePPN сервер авторизации определяет организацию — клиента, а по ней - Сервер аутентификации и учёта проекта.

Далее (2) UA пользователя для ввода пароля перенаправляется Сервером авторизации СКЦ к Серверу аутентификации и учёта проекта. В обращении (2) в качестве дополнительного атрибута передаётся имя учётной записи (login), которое будет использовано для входа пользователя на суперкомпьютер. Имя учётной записи (login) определяется на основе eduPersonPrincipalName, может быть заведён ранее, может быть сгенерирован динамически и запомнен вместе с eduPersonPrincipalName для дальнейшего использования, может быть сгенерирован на время прохождения задачи.

После ввода пароля пользователем (3) и подтверждения идентификации пользователя Сервером аутентификации и учёта, результат аутентификации (AUTH_STATUS) сообщается (4) Серверу авторизации СКЦ. В сообщении также указывается проект и договор (project), в рамках которого пользователь запускает задание (или задания), а также выделенная для этого квота расходования вычислительных ресурсов или

набор квот по видам ресурсов (quota).

Сервер авторизации создаёт или разблокирует (5) учётную запись login на вычислительном ресурсе (сервере доступа суперкомпьютера), связывает её с проектом project, записывает публичный ключ pkey в файл .ssh/authorized_keys домашнего каталога учётной записи login на сервере доступа. Кроме того, системе управления заданиями передаётся информация о квоте вычислительных ресурсов (quota), доступных пользователю в рамках проекта. Также в системе учёта может использоваться информация о eduPersonPrincipalName (ePPN).

Параллельно UA получает (6) ответ о результатах авторизации на ресурсе, после чего вычислительный ресурс доступен пользователю для работы по протоколу SSH (7).

При обмене сообщениями между сервером аутентификации и учёта организации - клиента и сервером авторизации СКЦ используются дополнительные атрибуты по сравнению со стандартным набором атрибутов при обмене сообщениями между провайдером идентификации (в терминах удостоверяющих федераций) IdP (Identity Provider) и сервис - провайдером SP (Service Provider) в удостоверяющей федерации [15]. Сервер аутентификации и учёта проекта для корректной обработки персональных данных пользователей должен находится именно в организации — клиенте, однако его развёртывание и поддержка является более сложной задачей, чем развёртывание «из коробки» и поддержка стандартного сервера аутентификации «универсальной» удостоверяющей федерации, например, удостоверяющей федерации RUNNetAAI ФГУ ФНЦ НИИСИ РАН [35], реализованной в Национальной исследовательской компьютерной сети (НИКС). Это может стать препятствием для широкого использования федеративного подхода к аутентификации и авторизации в сети СКЦ.

Выход видится в применении промежуточного слоя «управления проектом» между сервером аутентификации удостоверяющей федерации RUNNetAAI (IdP) в организации - клиенте и сервером авторизации (SP) суперкомпьютерного центра (см. рис. 2).

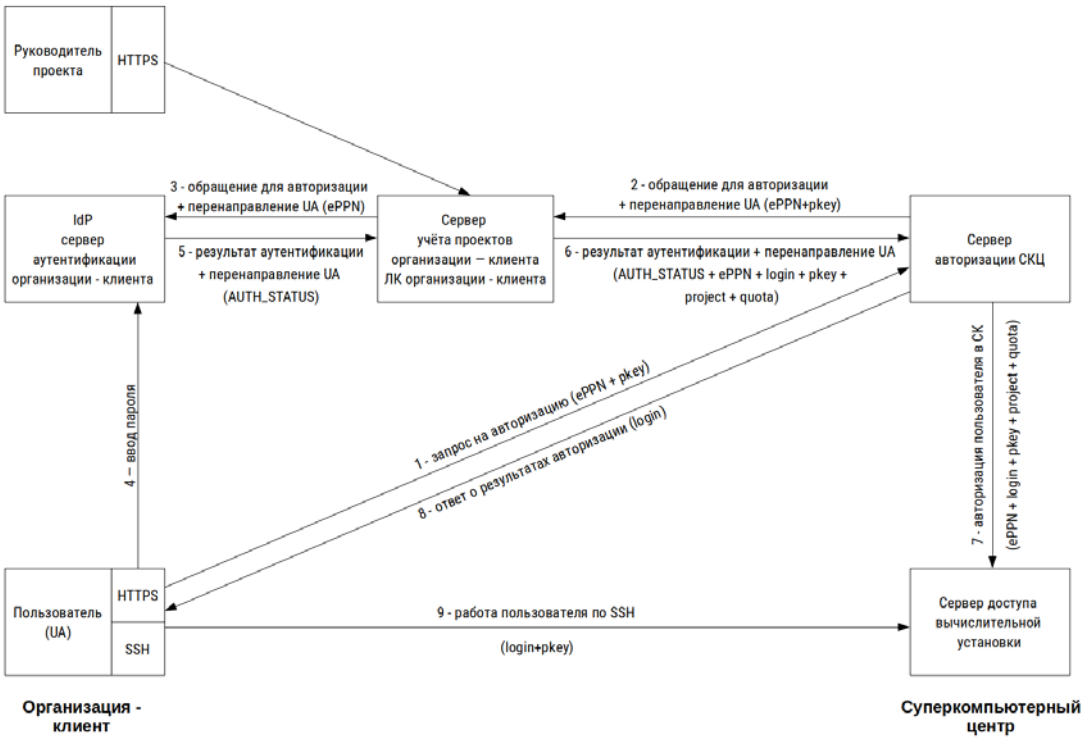


Рис. 2. Авторизация в сети СКЦ на основе федеративной аутентификации и сервера учёта проектов организации

В рассматриваемом случае запрос пользователя на авторизацию перенаправляется (2) с Сервера авторизации СКЦ на сервер управления проектами организации. Аутентификации пользователя производится в ходе стандартной процедуры идентификации удостоверяющей федерации RUNNetAAI: запрос с

eduPersonPrincipalName (ePPN) направляется (3) на сервер IdP от сервера учёта, после ввода пароля (4) результат аутентификации возвращается (5) на сервер управления проектами. Именно сервер управления проектами определяет учётное имя login для доступа к ресурсу, связывает его с eduPersonPrincipalName, определяет проект (project) и квоты (quota). Поскольку сервер управления проектами организации не манипулирует персональными данными пользователей организации, он может быть реализован как служба сети СКЦ.

В этом случае сервер управления проектами организации является частью общей распределенной службы сети СКЦ, которая поддерживает распределенную базу данных, доступную для всех систем управления заданиями всех СКЦ сети. Такая база обеспечит сопоставление идентификаторов и учётных записей пользователей, проектов, квот и договоров — всех данных, необходимых для планирования обработки заданий в сети СКЦ.

Взаимодействие организации — клиента с базой осуществляется через личный кабинет организации в сети СКЦ. Это отменяет необходимость развёртывания и поддержки специализированного сервера аутентификации, совмещённого с сервером учёта, и значительно снижает сложность применения федеративной аутентификации в организациях — клиентах.

4. Заключение

Результатом работы является уточнение процедур информационного обмена и состава информации (метаданных), предоставляемых организацией — клиентом суперкомпьютерному центру при обработке вычислительных заданий и передаваемых при перераспределении задания внутри сети СКЦ. Определено понятие «проект» как сущность системы авторизации и учёта. Предложена схема распределенной авторизации в сети СКЦ на основе удостоверяющей федерации RUNNetAAI, развёрнутой в Национальной исследовательской компьютерной сети и распределенной службы управления проектами организаций — клиентов сети СКЦ. Предложенный метод значительно упрощает внедрение и применение федеративной системы аутентификации и авторизации для организаций — клиентов распределенной сети СКЦ.

Благодарности. Исследования проведены при поддержке РФФИ, грант № 19-07-01223. При проведении исследований использовались суперкомпьютеры МСЦ РАН: МВС-10П и МВС-10П ОП.

Some Issues of Federated Authentication in a Distributed Network of Supercomputer Centers

Al.A. Gonchar, Yu.N. Morin, A.P. Ovsyannikov

Abstract. The article discusses the issues of federated approach for authentication, authorization and accounting in a distributed network of supercomputer centers. The article considered some scenarios for processing computing jobs, and the metadata provided by the client organization to the supercomputer center for computing jobs processing and redistributing within the network. The concept of "project" is defined as the essence of the authorization and accounting system. The scheme of distributed authorization in the distributed network of the supercomputer centers is proposed based on the identity federation RUNNetAAI deployed in the Russian National research computer network and the distributed project management service of client organizations. The proposals simplify the use of a federated authentication and authorization for organizations - clients of the distributed network of the supercomputer centers.

Keywords: distributed network of the supercomputer centers, AAA, authentication, authorization and accounting, identity federation, RUNNetAAI

Литература

1. А.В. Баранов, В.В. Молоканов, П.Н. Телегин, А.И. Тихомиров. Применение метода английского аукциона при планировании заданий с абсолютными приоритетами в распределенной вычислительной системе. «Программные продукты и системы», Т. 31 (2018.), № 3, 461–468, DOI: 10.15827/0236-235X.031.3.461-468

2. Г.И. Савин, Б.М. Шабанов, А.В. Баранов, А.П. Овсянников, А.А. Гончар. Об использовании федеральной научной телекоммуникационной инфраструктуры для суперкомпьютерных вычисле-

ний. «Вестник Южно-Уральского государственного университета». Серия: «Вычислительная математика и информатика», Т. 9 (2020), № 1, 20-35. DOI: 10.14529/cmse200102

3. Б.М. Шабанов, А.П. Овсянников, А.В. Баранов, С.А. Лещев, Б.В. Долгов, Д.Ю. Дербышев. Проект распределенной сети суперкомпьютерных центров коллективного пользования. «Программные системы: теория и приложения», 2017. № 4 (35), 245–262. DOI: 10.25209/2079-3316-2017-8-4-245-262

4. А.В. Баранов, А.И. Тихомиров. Методы и средства организации глобальной очереди заданий в территориально распределенной вычислительной системе. «Вестник Южно-Уральского государственного университета». Серия: «Вычислительная математика и информатика», Т. 6 (2017), № 4, 28-42. DOI: 10.14529/cmse170403

5 А.В. Баранов, А.И. Тихомиров. Планирование заданий в территориально распределенной системе с абсолютными приоритетами. «Вычислительные технологии», Т. 22 (2017), № S1, 4–12.

6. A.Baranov, P.Telegin, A.Tikhomirov. Comparison of Auction Methods for Job Scheduling with Absolute Priorities. “Lecture Notes in Computer Science”, vol. 10421 (2017). pp. 387-395. DOI: 10.1007/978-3-319-62932-2_37

7. B.M.Shabanov, P.N.Telegin, O.S.Aladyshev, A.V.Baranov, A.I.Tikhomirov. Comparison of Priority-Based and First Price Sealed-Bid Auction Algorithms of Job Scheduling in a Geographically-Distributed Computing System. Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, ElConRus 2018. P. 1557-1562. DOI: 10.1109/EIConRus.2018.8317396

8. Б.М. Шабанов, О.И. Самоваров. Принципы построения межведомственного центра коллективного пользования общего назначения в модели программно-определяемого ЦО. «Труды Института системного программирования РАН», Т. 30 (2018), № 6, 7–24. DOI: 10.15514/ISPRAS-2018-30(6)-1.

9. Б.М. Шабанов, П.Н. Телегин, А.П. Овсянников, А.В. Баранов, А.И. Тихомиров, Д.С. Ляховец. Система управления заданиями распределенной сети суперкомпьютерных центров коллективного пользования. «Труды НИИСИ РАН», Т. 8 (2018), № 6, 65-73. DOI: 10.25682/NIISI.2018.6.0009

10. Baranov A., Lyakhovets D., Savin G., Shabanov B., Telegin P. Measure of adequacy for the supercomputer job management system model // Proceedings of the 2019 Federated Conference on Computer Science and Information Systems, FedCSIS 2019. 2019. С. 423-426. DOI: 10.15439/2019F186

11. С.А. Лещев, О.С. Аладышев. Особенности современных сетей компьютерной памяти в суперкомпьютерных центрах. «Труды НИИСИ РАН», Т. 7 (2017), № 4, 151-156.

12. Baranov A.V., Leshchev S.A. Methods and means of distributed storage systems implementation. “Software Journal: Theory and Applications”. 2019. № 3. С. 3. DOI: 10.15827/2311-6749.19.3.3

13. А.В. Баранов, Е.А. Киселёв, Е.С. Кормилицин, В.Ф. Огарышев, П.Н. Телегин. Модернизация подсистемы сбора и обработки статистики центра коллективного пользования вычислительными ресурсами МСЦ РАН. «Труды НИИСИ РАН», Т. 8. (2018), № 4, 136-144.
DOI: 10.25682/NIISI.2018.4.0016

14. А.В. Баранов, А.П. Овсянников, Б.М. Шабанов. Федеративная аутентификация в распределенной инфраструктуре суперкомпьютерных центров. «Труды НИИСИ РАН», Т. 8 (2018), № 6, 79–83. DOI: 10.25682/NIISI.2018.6.0011.

15. А.Г. Абрамов, И.В. Васильев, В.А. Порхачёв. Принципы функционирования и управления удостоверяющей федерацией RUNNetAAI в рамках интерфедеративного взаимодействия с проектом eduGAIN. «Информатизация образования и науки», 2019. № 2(42), 40-47.

16. А.В. Баранов, Е.А. Киселёв. Облачные сервисы для научных высокопроизводительных вычислений на базе платформы PROXMOX. «Вычислительные технологии», Т. 24 (2019), № 6, 5-12. DOI: 10.25743/ICT.2019.24.6.002

17. О.С. Аладышев, А.В. Баранов, А.П. Овсянников, Г.А. Балаян, В.С. Синицин. Методы и средства совмещения потоков заданий от облачных платформ и менеджеров управления ресурсами суперкомпьютера. «Программные продукты, системы и алгоритмы», 2018. № 4, 54-63. DOI: 10.15827/2311-6749.29.337

18. Е.А. Киселев, А.В. Баранов. Облачная среда для высокопроизводительных вычислений на базе платформы Proxmox. «V Международная конференция «Информационные технологии и высокопроизводительные вычисления» (ITHPC-2019), Россия, Хабаровск, 16-19 сентября 2019 г. Сборник трудов. Под ред. С.И. Смагин, А.А. Зацаринный.

19. А.В. Баранов, А.А. Зонов Вариант организации облачного сервиса для высокопроизводительных вычислений. «Программные системы: теория и приложения», Т. 3 (2016), № 7, 3–23. DOI: 10.25209/20793316-2016-7-3-3-23.

20. А.В. Баранов, Б.В. Долгов, А.В. Федотов. Контейнеризация пользовательских заданий в суперкомпьютерной системе коллективного пользования. «Труды НИИСИ РАН», Т. 9 (2019), № 6, 123-

131. DOI: 10.25682/NIISI.2019.6.0016

21. А.В. Баранов, Д.С. Николаев. Использование контейнерной виртуализации в организации высокопроизводительных вычислений. «Программные системы: теория и приложения», Т. 7 (2016), № 1 (28), 117–134. DOI: 10.25209/2079-3316-2016-7-1-117-134

22. Г.И. Савин, П.Н. Телегин, А.В. Баранов, А.С. Шитик. Способы и средства представления пользовательских суперкомпьютерных заданий в виде контейнеров Docker. «Труды НИИСИ РАН», Т. 8 (2018), № 6, 84-93. DOI: 10.25682/NIISI.2018.6.0012

23. Б.М. Шабанов, А.П. Овсянников, А.В. Баранов, О.С. Аладышев, Е.А. Киселёв, Я.О. Жуков. Методы управления параллельными заданиями суперкомпьютера, требующими развёртывания отдельных программных платформ и виртуализации сетей // В сб.: Суперкомпьютерные дни в России. Труды международной конференции. 2017, 616-627.

24. A.V. Baranov, G.I. Savin, B.M. Shabanov et al. Methods of Jobs Containerization for Supercomputer Workload Managers. “Lobachevskii Journal of Mathematics”, Vol. 40 (2019), No. 5, 525–534. DOI: 10.1134/S1995080219050020.

25. O.S. Aladyshev, A.V. Baranov, R.P. Ionin, E.A. Kiselev, B.M. Shabanov. Variants of deployment the high performance computing in clouds // Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, ElConRus 2018. 2018. С. 1453-1457. DOI: 10.1109/EIConRus.2018.8317371

26. А.А. Гончар, А.П. Овсянников, А.А. Сорокин, Б.М. Шабанов, А.В. Юрченко. Развитие федеральной телекоммуникационной и вычислительной инфраструктуры в интересах науки и образования. «Вычислительные технологии», Т. 24 (2019), № 6, 21-29. DOI: 10.25743/ICT.2019.24.6.004

27. А.В. Баранов, Д.В. Вершинин, Д.Ю. Дербышев, Б.В. Долгов, С.А. Лещев, А.П. Овсянников, Б.М. Шабанов. Об эффективности использования канала связи между территориально удаленными суперкомпьютерными центрами. «Труды НИИСИ РАН», Т. 7 (2017), № 4, 137–142.

28. XSEDE System Requirements Specification v3.1. URL: http://hdl.handle.net/2142/45102 (дата обращения: 30.09.2020).

29. AARC: Authentification and Authorisation for Research Collaborations [электронный ресурс] // URL: https://aarc-project.eu/ (дата обращения 30.10.2020)

30. EUDAT-PRACE pilot for cross-infrastructure access to resources // URL: https://wiki.geant.org/display/AARC/EUDAT-PRACE+pilot+for+cross-infrastructure+access+to+resources (дата обращения: 30.10.2020)

31. A.B.Yoo, M.A.Jette, M.Grondona. (2003) SLURM: Simple Linux Utility for Resource Management. In: Feitelson D., Rudolph L., Schwiegelshohn U. (eds) Job Scheduling Strategies for Parallel Processing. JSSPP 2003. Lecture Notes in Computer Science, vol 2862. Springer, Berlin, Heidelberg. pp. 44-60. DOI: 10.1007/10968987_3

32. Henderson R.L. Job scheduling under the Portable Batch System. In: Job scheduling strategies for parallel processing, Feitelson D.G., Rudolph L. (Eds.), LNCS, Springer, 1995, vol. 949, pp. 279–294. DOI: 10.1007/3-540-60153-8_34.

33. Moab HPC Suite Enterprise Edition. [электронный ресурс] // https://support.adaptivecomputing.com/products/hpc-products/moab-hpc-suite-enterprise-edition/ (дата обращения 30.10.2020)

34. Система управления прохождением параллельных заданий. Руководство программиста (пользователя), (2016). http://www.jscc.ru/wp-content/uploads/2017/06/SUPPZ-user-guide-2016.pdf (дата обращения: 30.10.2020)

35. Регламент Удостоверяющей Федерации RUNNetAAI Федерального государственного учреждения «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук» (ФГУ ФНЦ НИИСИ РАН) // URL: https://niks.su/documents-ru-2/policy-2 (дата обращения 04.11.2020)

Способ оценки энергопотребления заданий в суперкомпьютерной системе коллективного пользования

Е.А. Киселёв[1], В.И. Киселев[2], А.В. Баранов[3], О.С. Аладышев[4]

[1]Межведомственный суперкомпьютерный центр РАН, Москва, Россия, kiselev@jscc.ru;

[2]Российский университет транспорта (МИИТ), Москва, Россия, kiselev40@mail.ru;

[3]Межведомственный суперкомпьютерный центр РАН, Москва, Россия, antbar@mail.ru;

[4]Межведомственный суперкомпьютерный центр РАН, Москва, Россия, aladyshev@jscc.ru

Аннотация. В статье рассматривается способ оценки энергопотребления заданий пользователей суперкомпьютерных центров коллективного пользования, заключающийся в сравнении энергоэффективности планирования суперкомпьютерных заданий по данным энергопотребления и времени выполнения параллельных программ. Рассмотрены разработанные авторами программные средства, позволяющие определять профиль энергопотребления параллельной программы в автоматическом режиме без внесения изменений в ее исходный код. Приведены результаты сравнения энергопотребления тестовых программ NAS Parallel Benchmarks (BT, EP, IS, LU и SP) на вычислительных системах с микропроцессорами Intel семейства Broadwell, Cascade Lake, Knights Landing и Skylake.

Ключевые слова: энергопотребление, параллельные программы, суперкомпьютеры, NAS Parallel Benchmarks, Intel

1. Введение

Увеличение производительности современных суперкомпьютеров напрямую связано с увеличением количества входящих в их состав вычислительных устройств (микропроцессоров, графических ускорителей и специализированных сопроцессоров). Обратной стороной масштабируемости и высокой производительности вычислительной системы является ее энергопотребление. Существующие ограничения энергосетей суперкомпьютерных центров (СКЦ) стимулируют переход на более энергоэффективные решения, однако такое обновление происходит поэтапно: старые суперкомпьютеры выводят из эксплуатации по частям, постепенно замещая более новыми. Одновременно в СКЦ может функционировать нескольких вычислительных систем с разным энергопотреблением, как, например, в Межведомственном суперкомпьютерном центре РАН [1].

В работах [2, 3] показано, что для некоторых классов параллельных программ, например, с большим количеством фаз коммуникационного обмена или дисковых операций, энергопотребление может быть снижено без увеличения времени их выполнения. При этом для параллельных программ с преобладанием фаз вычислений, понижение тактовой частоты микропроцессора приводит к существенному увеличению времени выполнения и является недопустимым. Одни и те же программы, запущенные с одинаковыми входными параметрами на разных вычислительных системах, могут по-разному влиять на их энергопотребление. В таких условиях актуальной становится задача исследования профиля энергопотребления параллельных программ с целью планирования их запуска на суперкомпьютерных ресурсах с минимальным энергопотреблением (далее – энергоэффективное планирование).

Энергоэффективное планирование суперкомпьютерных ресурсов невозможно без программных средств, позволяющих в автоматическом режиме определять профиль энергопотребления выполняемой параллельной программы. Необходимо также по профилю энергопотребления программ пользователей определять энергоэффективность вычислительной системы. Перечисленным вопросам и посвящена настоящая работа, являющаяся продолжением исследований [4].

2. Методы и средства измерения энергопотребления вычислительных систем

Современные суперкомпьютеры, как правило, имеют кластерную архитектуру. В составе таких систем могут использоваться дисковые или бездисковые вычислительные узлы (ВУ),

укомплектованные многоядерными микропроцессорами и, в некоторых случаях, графическими ускорителями и специализированными сопроцессорами. Все ВУ объединены между собой и с подсистемой хранения высокоскоростными сетями.

В СКЦ измерение энергопотребления ВУ производится с помощью программных средств мониторинга вычислительных ресурсов. В работе [5] авторами было проведено экспериментальное сравнение подходов и методов измерения энергопотребления вычислительных систем. В работах [6-9] также рассматривались подходы и методологии измерения энергоэффективности вычислительных средств. Определено, что наиболее подходящими для применения в системах коллективного пользования являются следующие программные методы измерения энергопотребления.

1. Метод оценки энергопотребления на основе данных из MSR-регистров. На базе регистров MSR строятся аппаратные счетчики производительности HPC (от англ. Hardware Performance Counter) и RAPL- регистры (от англ. Running Average Power Limit). В счетчиках HPC хранится информация о различного рода событиях, возникающих во время работы программы: количество целочисленных операций, операций с плавающей точкой, операций обращения к памяти, количество кэш-промахов, тип выполняемой инструкции и др. В работах [8-11] показана зависимость между хранящимися в счетчиках HPC и RAPL-регистрах значениями и энергопотреблением ВС. Для доступа к регистрам MSR реализованы API, например, Libmsr API. Данный метод позволяет определить текущее энергопотребление микропроцессора.

2. Метод оценки энергопотребления по данным, предоставляемым ОС. Метод основан на анализе данных загрузки компонентов ВС [10] и анализе данных энергопотребления с помощью механизма ACPI (Advanced Configuration and Power Interface) [6], который предоставляет ОС интерфейс для управления питанием. Данный метод позволяет оценить вклад выполняющихся процессов в общую долю энергопотребления ВС.

3. Метод оценки энергопотребления графических ускорителей с использованием библиотеки NVML [12]. На текущий момент только компания NVIDIA предоставляет кроссплатформенную открытую библиотеку NVIDIA Management Library, где пользователям наиболее производительных видеокарт линейки Tesla, Quadro, Grid предоставляется доступ к данным о текущем энергопотреблении посредством специального API NVML.

Практическое применение всех перечисленных методов позволило выявить их преимущества и недостатки. Преимуществами 1-го и 3-го методов являются точность измерения, однако для их применения необходимо наличие специализированных библиотек, совместимых с установленными в ВУ микропроцессорами. Преимуществом 2-го метода является его универсальность. Опыт применения 2-го метода позволил сделать вывод, что его целесообразно использовать для оценки пиковых или значений общего энергопотребления всех компонентов системы (в этих случаях точность измерений такая же, как в 1-м и 3-м методах).

3. Программное средство автоматизированного построения профиля энергопотребления параллельных приложений пользователей

В СКЦ для запуска параллельных программ используются системы управления заданиями (СУЗ) [13]. Примерами таких систем являются Slurm [14], PBS [15] и др. В МСЦ РАН для этих целей более 20 лет применяется отечественная Система управления прохождением параллельных заданий (СУППЗ) [1]. СУЗ контролирует очередность доступа пользователей к суперкомпьютерным ресурсам, организует запуск и осуществляет контроль за ходом выполнения параллельных программ. Параметры запуска параллельных программ определяются пользователями через специальный командный интерфейс СУЗ. В некоторых СУЗ параметры запуска параллельной программы могут быть оформлены в виде отдельного файла паспорта задания. Запуск параллельных программ осуществляется в соответствии с определяемой СУЗ очередностью, при этом ВУ выделяются программе динамически из числа свободных на момент запуска. На одном ВУ, как правило, может выполняться только одно параллельное приложение.

Авторами реализован и интегрирован с СУППЗ программный модуль автоматического сбора и обработки данных об энергопотреблении параллельных программ. Идея предлагаемого решения состоит в следующем. В момент выделения параллельной программе вычислительных ресурсов на каждом из ВУ инициируется запуск программы-монитора. Программа-монитор ежесекундно собирает данные о текущем энергопотреблении ВУ. В качестве источников сбора данных об энергопотреблении могут выступать интерфейсы powercap, perf_event_open или регистров MSR. Выбор интерфейса сбора данных об энергопотреблении

осуществляется администратором СУППЗ. Сразу после завершения параллельной программы и перед освобождением ВУ, программа-монитор формирует для каждого ВУ отдельный файл с данными об энергопотреблении и времени выполнения параллельной программы, упаковывает результаты в архив и размещает их в служебном каталоге СУППЗ.

Обработка данных об энергопотреблении параллельного приложения осуществляется администратором СУППЗ через разработанную авторами программу-анализатор. Программа автоматически рассчитывает минимальное, максимальное и среднее энергопотребление каждого ВУ в момент выполнения программы, а также время выполнения параллельной программы.

4. Способ оценки профиля энергопотребления параллельных программ

На основе данных, полученных с помощью разработанных программных средств, были исследованы профили энергопотребления тестов NAS Parallel Benchmarks (NPB), запущенных на разделах установленного в МСЦ РАН суперкомпьютера МВС-10П ОП: KNL, Broadwell (BDL), Skylake (SLK) и Cascade lake (CLK) [16].

Для исследования были выбраны следующие тестовые задания NPB:

- EP – позволяет оценить производительность КВС при минимальном межпроцессорном взаимодействии;
- IS – позволяет оценить производительность сетевой подсистемы и подсистемы работы с общей памятью;
- BT – позволяет оценить производительность суперкомпьютера при выполнении операций ввода/вывода;
- LU – позволяет оценить влияние латентности коммуникационной среды на производительность суперкомпьютера;
- SP – позволяет оценить производительность суперкомпьютера при оптимальной загрузке сети.

Запуск каждого теста производился через СУППЗ. В таблице 1 приведено количество вычислительных ядер, необходимых для запуска каждого теста, и количество ВУ, выделенных СУППЗ для выполнения тестовых заданий.

Таблица 1. Параметры запуска тестов NPB

Название теста	**Кол-во ядер**	**Количество выделенных ВУ**			
		Broadwell	**CLK**	**KNL**	**Skylake**
BT	144	5	3	2	4
EP	144	5	3	2	4
IS	256	8	6	4	8
LU	256	8	6	4	8
SP	256	8	6	4	8

Полученные с помощью разработанных программных средств данные об энергопотреблении и времени выполнения параллельных заданий позволили:

- определить ВУ, на которых достигаются наименьшие энергозатраты и наименьшее время выполнения (таблица 1, рис. 1);
- оценить разницу в энергозатратах между заданиями с минимальным энергопотреблением и минимальным временем выполнения (Таблица 2, рис. 2).

В таблице 2 названия разделов Broadwell, Cascade Lake и Skylake сокращены до BDL, CLK и SLK соответственно.

Таблица 2. Количество энергии (кДж), затраченной на выполнение тестов NPB, и времени их выполнения на вычислительных узлах Broadwell, Cascade lake, KNL, Skylake

Тест	**Энергозатраты (кДж)**			
	BDL	**CLK**	**KNL**	**SLK**
BT	265,91	227,34	239,30	282,24
EP	30,89	31,59	39,29	37,30
IS	15,87	13,76	8,23	13,22
LU	172,42	172,30	6,87	188,01
SP	335,24	294,46	235,80	361,18
	Время выполнения (с)			
BT	215,68	172,61	623,83	169,93
EP	24,76	25,18	132,95	23,50
IS	5,07	3,51	6,28	3,48
LU	81,42	71,72	316,45	64,90
SP	160,18	118,95	317,72	114,24

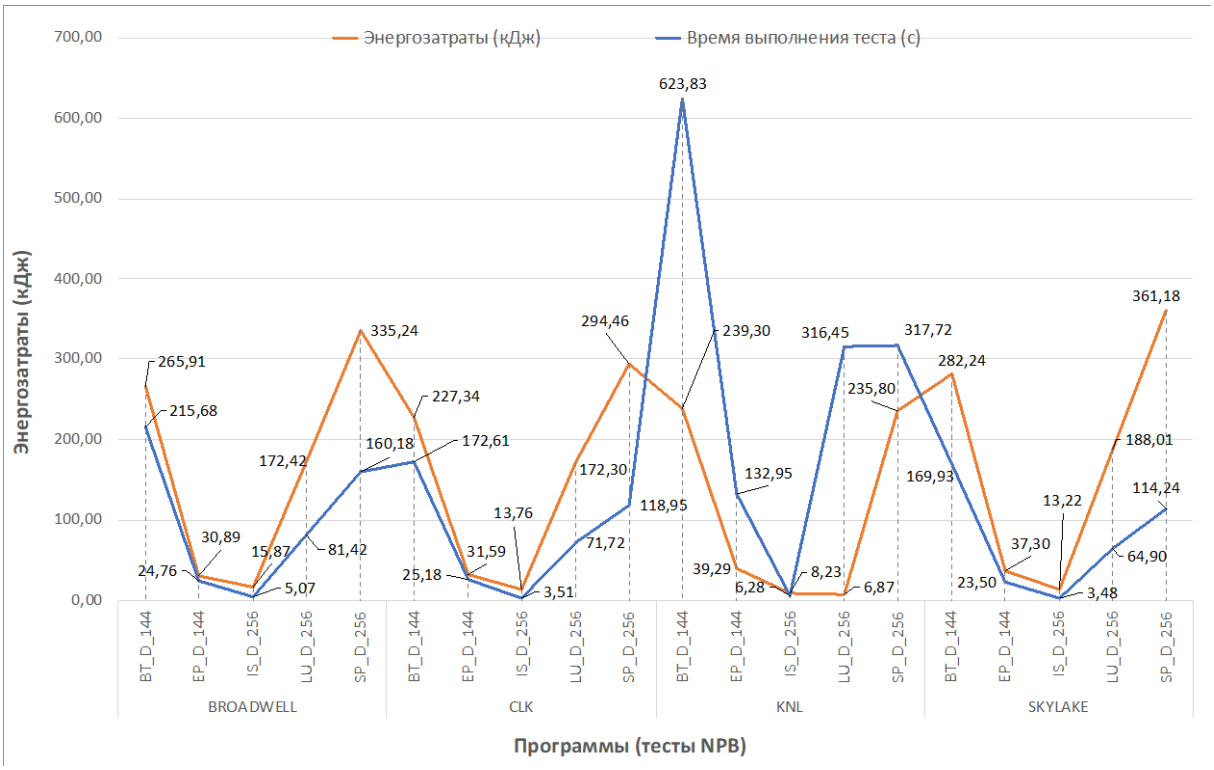


Рис. 1. Количество энергии (кДж), затраченной на выполнение тестов NPB, и времени их выполнения на вычислительных узлах Broadwell, Cascade lake, KNL, Skylake

Таблица 3. Энергозатраты на задания с минимальным энергопотреблением и минимальным временем выполнения

Тесты NPB (Class D)	**BT**	**EP**	**IS**	**LU**	**SP**
Энергозатраты (при минимальном времени выполнения теста), кДж	282,24	37,30	13,22	188,01	361,18
Минимальные энергозатраты, кДж	227,34	30,89	8,23	6,87	235,80
Разница в энергозатратах, %	19,45	17,18	37,74	96,35	34,71
Время выполнения теста (при минимальных энергозатратах), с	172,61	24,76	6,28	316,45	317,72
Минимальное время выполнения теста, с	169,93	23,50	3,48	64,90	114,24
Разница во времени выполнения (%)	1,55	5,09	44,59	79,49	64,04

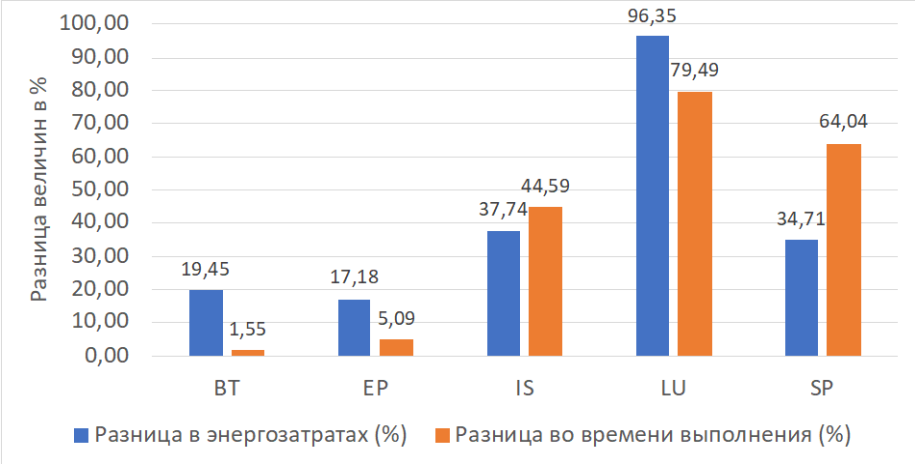


Рис. 2. Результаты сравнения энергозатрат на задания с минимальным энергопотреблением и минимальным временем выполнения

Анализ изменения энергопотребления ВУ при выполнении параллельных программ позволяет оценить характерные пиковые значения энергопотребления. Представленные на рисунке 2 результаты позволяют сделать вывод, что для тестов BT и EP целесообразно перераспределение на более энергоэффективные суперкомпьютерные системы, например, Skylake, при незначительном увеличении времени выполнения (1,55% и 5,09% соответственно). Для тестов IS, LU и SP сокращение энергопотребления возможно при увеличении времени выполнения и наоборот.

5. Заключение

Задача исследования профиля энергопотребления параллельных программ имеет важное значение для организации энергоэффективного планирования заданий в суперкомпьютерных системах коллективного пользования. В работе предложен способ оценки энергопотребления заданий в суперкомпьютерной системе коллективного пользования, заключающийся в сравнении энергоэффективности планирования заданий по данным энергопотребления и времени выполнения параллельных программ. С той целью в состав системы управления заданиями суперкомпьютера были интегрированы специально разработанные программные средства, позволяющие в автоматическом режиме собирать данные о профиле энергопотребления параллельных программ без внесения изменений в их исходный код.

Предложенный способ был применен для оценки энергопотребления заданий, выполнявшихся на разделах установленного в МСЦ РАН суперкомпьютера МВС-10П ОП под управлением системы коллективного пользования СУППЗ. В качестве параллельных программ были использованы тесты BT, EP, IS, LU и SP из набора NAS Parallel Benchmark. Полученные результаты демонстрируют, что предложенный способ оценки энергопотребления может быть использован при исследовании методов, способов и алгоритмов энергоэффективного планирования вычислительных ресурсов суперкомпьютера с целью сокращения их энергопотребления.

Исследования проведены при поддержке РФФИ (грант № 19-07-01072).

The Power Consumption Estimation of Super-Computer Jobs

E. Kiselev, V. Kiselev, A. Baranov, O. Aladyshev

Abstract. The article discusses the power consumption estimation method of supercomputer user jobs in supercomputer centers. The method consists in comparing power consumption data on scheduling supercomputer jobs based and on jobs execution time. The authors consider software tools that allow automatic determination the parallel program power consumption profile without source code changes. The comparison results of the NAS Parallel Benchmarks (BT, EP, IS, LU, and SP) power consumption on supercomputer MVS 10P OP installed at the JSCC RAS with Broadwell, Cascade Lake, Knights Landing and Skylake Intel microprocessors are presented.

Keywords: HPC, supercomputer, power consumption, NAS Parallel Benchmark, job scheduling, Intel

Литература

1. G. Savin, B. Shabanov, P. Telegin, A. Baranov. Joint Supercomputer Center of the Russian Academy of Sciences: Present and Future. “Lobachevskii Journal of Mathematics”, V. 40 (2019), N. 11, 1853–1862, https://doi.org/10.1134/S1995080219110271.

2. Y. Chen, S. Alspaugh, D. Borthakur, R. Katz. Energy efficiency for large-scale MapReduce workloads with significant interactive analysis. “7th ACM european conference on Computer Systems (EuroSys’12)”, USA, New York, 2012, 43–56, https://doi.org/10.1145/2168836.2168842.

3. N. Tiwari, S. Sarkar, U. Bellur, M. Indrawan. An Empirical Study of Hadoop's Energy Efficiency on a HPC Cluster. “Procedia Computer Science”, V. 29 (2014), 62-72, https://doi.org/10.1016/j.procs.2014.05.006.

4. E. Kiselev, V. Kiselev, B. Shabanov, O. Aladyshev, A. Baranov. The Energy Efficiency Evaluating Method Determining Energy Consumption of the Parallel Program According to its Profile. “Lobachevskii Journal of Mathematics”, V. 41 (2020), N. 12, 2542–2551, https://doi.org/ 10.1134/S1995080220120161.

5. Е.А. Киселев, А.В. Баранов, С.А. Лещев. Сравнительный анализ подходов и методов измерения энергопотребления вычислительных систем. «V Международная научно-практическая конференция «Информационные технологии и высокопроизводительные вычисления», Россия, Хабаровск, 2019, 66-71.

6. A. Noureddine, R. Rouvoy, L. Seinturier. A review of energy measurement approaches. “Operating Systems Review”, V. 47 (2013), I. 3, 42-49. https://doi.org/10.1145/2553070.2553077.

7. C. Lively, V. Taylor, X. Wu, H. Chang, C. Su, K. Cameron, S. Moore, D. Terpstra. E-AMOM: An Energy-Aware Modeling and Optimization Methodology for Scientific Applications on Multicore Systems. “Comp. Sci. – Res. and Dev.”, V. 29 (2014), I. 3, 197-210, https://doi.org/10.1007/s00450-013-0239-3.

8. S. Walker, M. McFadden. Best Practices for Scalable Power Measurement and Control. “IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)”, USA, Chicago, 2016, 1122–1131, https://doi.org/10.1109/IPDPSW.2016.91.

9. X. Wu, V. Taylor. Utilizing Hardware Performance Counters to Model and Optimize the Energy and Performance of Large Scale Scientific Applications on Power-Aware Supercomputers. “IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)”, USA, Chicago, 2016, 1180–1189, https://doi.org/10.1109/IPDPSW.2016.78.

10. C. Lively, X. Wu, V. Taylor, S. Moore, H.-C. Chang, K. Cameron. Energy and performance characteristics of different parallel implementations of scientific applications on multicore systems. “The International Journal of High Performance Computing Applications”, V. 25 (2011), I. 3, 342–350, https://doi.org/10.1177/1094342011414749.

11. D. Li, B. de Supinski, M. Schulz, K. Cameron, D. Nikolopoulos. Hybrid MPI/OpenMP power-aware computing. “2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)”, USA, Atlanta, 2010, 1–12, https://doi.org/10.1109/IPDPS.2010.5470463.

12. NVIDIA Management Library (NVML), https://developer.nvidia.com/nvidia-management-library-nvml.

13. A. Reuther. Scalable system scheduling for HPC and big data. “Journal of Parallel and Distributed Computing”, V. 111 (2018), 76–92, https://doi.org/10.1016/j.jpdc.2017.06.009.

14. A.B. Yoo, M.A. Jette, M. Grondona. SLURM: Simple Linux Utility for Resource Management. “Lecture Notes in Computer Science”, V. 2862 (2003), 44–60, https://doi.org/10.1007/10968987_3.

15. R.L. Henderson. Job scheduling under the Portable Batch System. “Lecture Notes in Computer Science”, V. 949 (1995), 279-294. https://doi.org/10.1007/3-540-60153-8_34.

16. Вычислительные ресурсы МСЦ РАН, http://www.jscc.ru/resources/hpc/.

Разработка топологии тестового фотошаблона для отработки технологического процесса формирования вольфрамовой металлизации в КМОП СБИС посредством метода химико-механической планаризации

С.В. Букатин[1], А.А. Столяров[2]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, Sergey_Bukatin@srisa.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, Alexander_Stolyarov@srisa.ru

Аннотация. Разработан новый принцип формирования топологии на примере тестовых структур для отработки технологического процесса формирования вольфрамовой металлизации с использованием химико-механической планаризации. Разделение повторяющихся структур на отдельные модули позволяет разместить всю необходимую топологию на меньшем количестве фотолитографических шаблонов, а также в дальнейшем дополнять или повторно использовать набор шаблонов для других тестовых структур.

Ключевые слова: тестовый шаблон, вольфрамовая металлизация, химико-механическая планаризация

1. Введение

При формировании кремниевых СБИС, нацеленных на эксплуатацию в расширенном температурном диапазоне, а именно от минус 60 до плюс 300°С, традиционными способами исполнения являются использование КНИ подложек, что позволяет снизить токи утечки, увеличить показатели по отказоустойчивости при повышенных температурах, и применение вольфрама(W) в качестве основного материала для формирования системы металлических слоев в структуре многоуровневой металлизации. Последнее обеспечивает возможность сохранения функционирования схем в расширенном температурном диапазоне, нежели при использовании системы металлических слоев на основе алюминия(Al). Это обуславливается высокой температурой плавления W пленки и низким значением коэффициента линейного расширения, а также вольфрамовая металлизация исключает электромиграцию при повышенных температурах, и как следствие, высокую надежность и срок службы изделий при высоких температурах.

В ранее выполненной работе [1] было показано, что традиционная конструкция транзисторов, используемая при проектировании КМОП СБИС устройств на уровне технологии 0,5 мкм, позволяет обеспечить необходимые требования по устойчивости выходных транзисторных характеристик в рассматриваемом диапазоне температур.

В то же время разработка операций формирования шин металлизации на основе W, требует применения иных подходов, создавая необходимость использования дополнительного обрабатывающего оборудования. В частности, при формировании шин металлизации в работе [2] был использован метод плазменного травления шин W по фоторезистивной маске, аналогичный методу формирования шин Al металлизации. Очевидным преимуществом такого подхода является его унификация по порядку изготовления с традиционными технологическими подходами, используемыми при формировании Al металлизации (рис.1, а). Однако недостатком метода следует признать применение дополнительного обрабатывающего оборудования в виде камер плазменного травления и, в ряде случаев, жидкостной химической очистки.

Альтернативным методом формирования шин W металлизации может считаться тот, при котором задействуется метод химико-механической планаризации (рис.1, б). Этот способ нашел без-альтернативное на сегодняшний день применение при формировании контактных областей в многоуровневых СБИС, а также при формировании межсоединений на основе меди (Cu) для технологических процессов 0,13 мкм и ниже[3].

Очевидно, что последний из описанных методов формирования системы W металлизации

нуждается в более тщательном изучении в части получения электрических характеристик металлизации, отработки правил проектирования, описания конструктивно-технологических приемов и т.п. В связи с этим необходима разработка специализированного тестового кристалла для изучения и отработки последовательности и режимов технологических операций для формирования шин W металлизации.

Традиционно, для отработки режимов операций базовых технологических процессов, используются специальные тестовые кристаллы [4], содержащие необходимо-достаточный набор тестовых структур, характеристики которых используются для принятия решений по правилу формирования конструктива, правилам взаимного расположения элементов проектных слоев, из которых формируется СБИС. Для исследования характеристик и формирования требований к изготовлению элементов СБИС с W металлизацией был спроектирован тестовый кристалл. При проектировании тестового кристалла преследовалось решение двух важных задач, а именно:

- исследование и разработка требований к формированию шин W металлизации в многоуровневых СБИС;

- экономия площади фотошаблона, позволяющая оптимизировать финансовые затраты на изготовление оснастки и проведение исследований.

В данной работе будет продемонстрирован подход к проектированию топологии тестового кристалла, позволяющий решить указанные задачи, а также будет дано описание разработанного тестового кристалла.

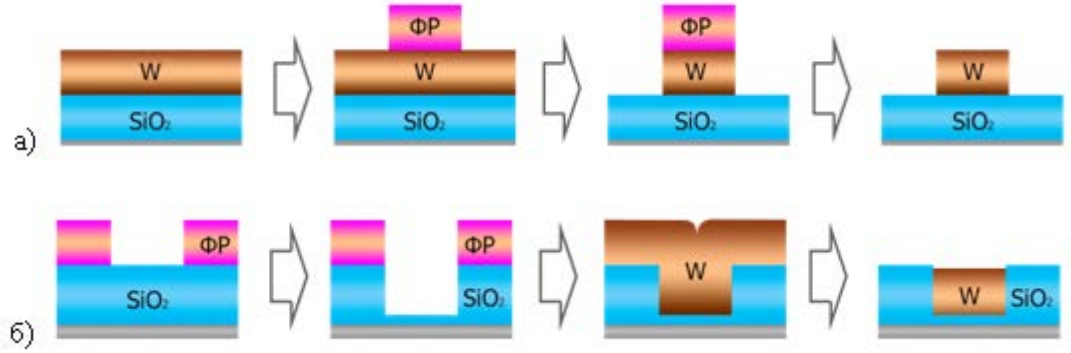


Рис. 1 Методы формирования металлизации: а)с использованием плазмо-химического травления, б) с использованием процесса химико-механической планаризации.

2. Варианты исполнения тестовых структур

При формировании тестовых структур необходимо заложить ряд тестов для оценки всех факторов процесса ХМП и дальнейшей характеризации технологического процесса формирования металлической разводки и разработки требований к проектированию.

Один из факторов побочного влияния процесса ХМП является различная скорость полировки по пластине. Как следствие – различная толщина итоговых шин и сопротивление шин металлизации от кристалла к кристаллу.

Вторым фактором является эрозия структур, из-за различной плотности заполнения области кристалла, что будет влиять на различие толщины и сопротивление шин металлизации внутри кристалла. В областях с высокой плотностью заполнения конечная толщина шин будет меньше, чем в областях с низкой плотностью заполнения, и, соответственно, в областях с высокой плотностью заполнения сопротивление шин будет выше.

В результате нам необходимо определить разброс сопротивления вольфрамовых шин, от пластины к пластине, по пластине и по кристаллу, и выбрать оптимальные параметры для минимизации негативного влияния процесса ХМП – шаг и геометрические размеры шин, шаг между шинами, максимально допустимое количество элементов на единицу площади.

Т.о. требуемые для проведения работы тестовые структуры:

- структуры для измерения сопротивления и геометрии одиночных шин различного размера (0,5, 0,7 и 1,0 мкм)

- структуры для измерения сопротивления и геометрии массива шин различного размера (0,5, 0,7 и 1,0 мкм) с различным расстоянием между шинами(0,5, 0,7 и 1,0 мкм)

- структуры для измерения тока в массиве шин

- структура для измерения сопротивления

одиночного переходного контакта и цепочки переходных контактов

- структуры для измерения геометрии и сопротивления широких шин (5-75 мкм)

Измерения профиля полировки в кристалле будет проводиться с помощью профилометра Dektak V200SL. Измерение толщины диэлектрических слоев на спектрофотометре Nanospec 8000XSE и Apecs 3020. А измерения электрических характеристик, сопротивления и токов утечки в структурах на аппаратно-программном комплексе на основе оборудования фирмы Keysight Technologies.

Для всех тестовых структур предлагается общая конструкция базовых элементов. Тестовый сегмент выполнятся в области размером 2,6 x 2,6 мм. Для всех сегментов кристалла является общим расположение контактных площадок. Две группы контактных площадок расположены в нижней и верхней части тестового сегмента. Группа состоит из двух рядов по 20 контактных площадок. Размер каждой площадки 60 x 60 мкм. Расстояние между контактами в горизонтальном направлении 60 мкм, в вертикальном направлении 60 мкм.

Дальнейшая работа предполагает выбор оптимальной схемы совмещения при подготовке образцов. Необходимо предусмотреть варианты для глобальных и сегментированных знаков совмещения. С этой целью по периметру сегмента располагается область для размещения меток совмещения и контроля рассовмещения для фотолитографического оборудования (рис. 2).

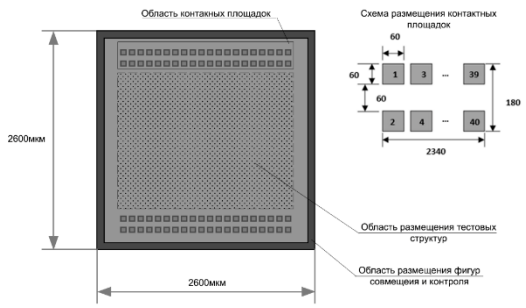


Рис. 2. Схема тестового сегмента

2.1. Структуры для исследования сопротивления одиночных и групповых шин металлизации

Первыми в тестовом шаблоне необходимо заложить структуры для характеризации одиночных и групповых шин вольфрама (рис. 3). При этом необходимо исследовать структуры с различной плотностью заполнения, сочетая различную ширину вольфрамовых шин “L” (0,5 мкм, 0,7 мкм, 1 мкм) и диэлектрическим зазором между шинами “S”.

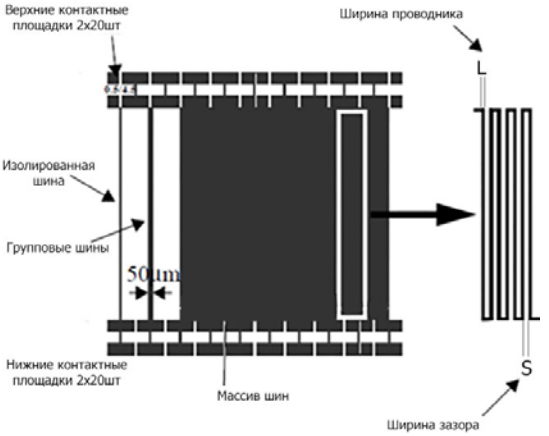


Рис. 3 Структуры для характеризации шин вольфрамовой металлизации

На тестовой области с массивом шин возможно оценить влияние эрозии структур после процесса полировки на сопротивление металлизации в областях с различной плотностью заполнения в кристалле. Оптимальные значения для сопротивления вольфрамовых шин должны находиться в диапазоне 0,15-0,25 Ом/квадрат. Так же в данных структурах закладываются тесты для измерения токов утечки. По данному параметру оценивается необходимая длительность полировки, не допускающая остатков металла между шинами. При отсутствии несполированного металла ток утечки должен быть менее 10-10 А.

2.2. Тестовые структуры с переходными контактами

В шаблоне также закладываются структуры для оценки качества формирования межслойных одиночных контактов и цепочек контактов (рис. 4). Ряд тестов проектируется с преднамеренным смещением контактов, для оценки влияния рассовмещения при проведении фотолито-графических операций на качество формируемой многослойной металлизации.

На данных структурах будет оцениваться целостность цепочек контактов (отсутствие разрыва структур при штатном совмещении и предусмотренном рассовмещении).

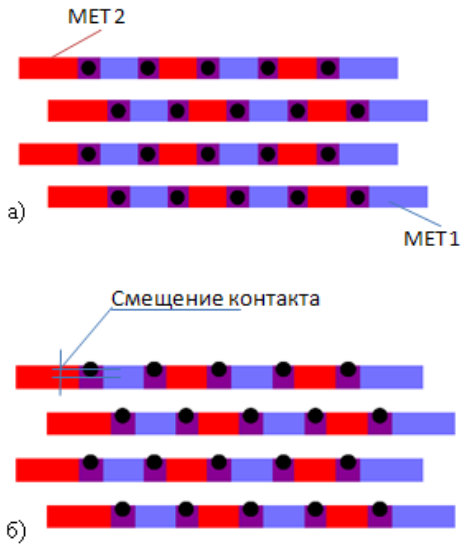


Рис. 4. Структуры для характеризации переходных контактов. а) без смещения VIA; б) со смещением VIA

2.3. Структуры для исследования характеристик широких шин

Одним из факторов формирования вольфрамовых шин, является ограничение на максимальную толщину осаждения тонких пленок вольфрама, т.к. такие пленки создают большое механическое напряжение пластины, что при формировании многослойной металлизации может привести в дальнейшем к проблемам при проведении процессов экспонирования фоторезистивных покрытий из-за ограниченности поля фокусировки. Так же возможно повреждение структур КМОП устройств и даже разрушение пластины из-за высоких механических напряжений. Таким образом ограничение толщины вольфрамовой пленки в 0,5 мкм накладывает ограничение на ширину металлизации <1 мкм (рис. 5).

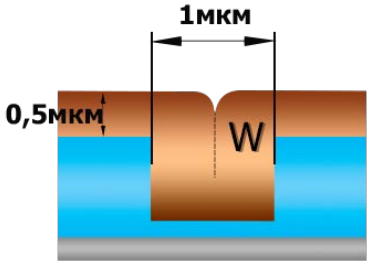


Рис. 5. Заполнение углублений в диэлектрике при осаждении пленок вольфрама толщиной 0,5 мкм

Требуется исследовать различные варианты формирования широких шин, для обеспечения неразрывности структуры металла. В широком проводнике формируются вырезы, заполненные диэлектриком, при этом уменьшается негативное влияние процесса ХМП ввиду уменьшения переполировки вольфрама в центре широкой шины, и позволяет использовать толщину пленки вольфрама 0,5 мкм (рис. 6).

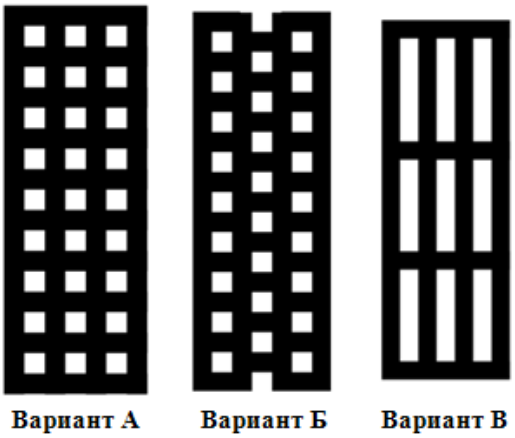


Рис. 6. Варианты исполнения широких шин металлизации

3. Разработка фотошаблона для тестовых структур

Так для проведения всех тестов необходимо сформировать структуры минимум 5 слоев:

- 2 слоя вольфрамовой металлизации для основных тестов (MET1W и MET2W)
- 1 слой металлизации для формирования контактных площадок (MET3)
- 2 слоя переходных контактов между 1-м – 2-м металлом, 2-м – 3-м металлом (VIA1, VIA2)

Основная идея подхода закладывается в формировании модульной структуры тестового шаблона. Базовым слоем формируются уникальные структуры, необходимые для формирования принципиальных тестовых структур, а вспомогательные слои (межслойные переходные окна, контактные площадки для измерений, вспомогательные соединения верхнего уровня и т.д.) мультиплицируются поверх.

Для представления создаются координатные сетки. Одна опорная координатная сетка, определяется размером минимального тестового модуля 2,6 х 2,6 мм. И большая сетка, кратная минимальной, определяемая количеством элементов в базовом слое. В работе принят размер большой сетки 5 х 5 опорных модулей (13 х 13 мм) (рис. 7).

Использование предложенного подхода реализации тестового шаблона имеет широкий ряд преимуществ:

1. Возможность разместить большее количество тестовых структур для одного проекта.

2. Уменьшение общего количества необходимых ФШ, вследствие оптимизации дублирующихся структур до одного базового модуля.

3. Потенциальная возможность для масштабирования модульной структуры, и применение части масок для других тестовых проектов.

4. Возможность дальнейшего развития и масштабирования созданной модульной структуры, посредством добавления новых базовых структур.

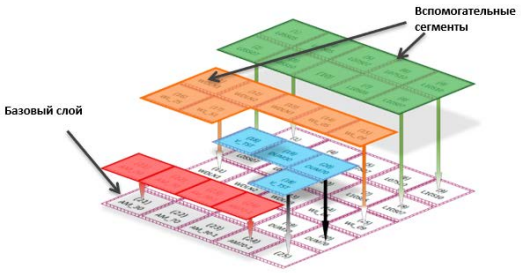


Рис. 7. Схема формирования тестовых структур с использованием базового слоя и вспомогательных модулей

Но при реализации, следует учитывать дополнительные факторы и особенности:

1. Изменение подхода к разработке топологии, требуется принять ряд правил проектирования топологии, чтобы обеспечить геометрическую совместимость различных модулей (точная область расположения контактных площадок, размеры базового модуля и т. д.).

2. Усложнение процесса трассировки топологии на масочные слои. Вводятся дополнительные проверки на целостность электрических цепей, проверка сходимости и эквивалентности структур. Проверка совмещения топологии вспомогательных модулей с остальными частями тестовой топологии.

3. Разработка более сложных программ экспонирования, где следует учесть экспонирование только в требуемых областях и т.д.

4. Для фотолитографического оборудования требуется разработка большего количества программ экспонирования, обеспечивающих совмещение опорной/малой и большой/производной сетки экспонирования. Определить области экспонирования для каждой вспомогательной структуры.

5. Необходимость применения мультиэкспонирования. Для создания топологии на пластине, требуется экспонирование по различным базовым шаблонам, в разных областях тестовых структур (рис. 8).

Рис. 8. Карта экспонирования в кристалле для различных функциональных слоев

4. Заключение

В данной работе при подготовке тестовых структур для исследования технологии формирования вольфрамовой металлизации, была проработана новая концепция подготовки модульного фотошаблона, что позволило разместить

все необходимые масочные слои тестовых структур на одном стекле фотошаблона, вместо пяти и более. Данный подход так же позволяет в дальнейшем расширять набор тестовых структур или использовать уже имеющиеся структуры в иных экспериментальных работах.

С использованием рассмотренного в статье шаблона, в дальнейшем будут проведены исследовательские работы по изучению характеристик вольфрамовой металлизации. По результатам чего будут подобраны оптимальные геометрические размеры для структур, плотность расположения элементов для разработки правил проектирования, обеспечивающих воспроизводимые электрофизические характеристики металлизации для имеющегося технологического процесса.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН (выполнение фундаментальных научных исследований) по теме № 0065-2019-0018 «Исследование и построение моделей и конструкций элементов микроэлектроники в расширенном диапазоне температур (-60…300°С)».

Development of a Test Photomask Topology for Testing the Technological Process of Forming Tungsten Metallization in CMOS VLSI Using the Chemical Mechanical Polishing

Bukatin S.V., Stolyarov A.A.

Abstract. A new principle of topology formation has been developed on the example of test structures for testing the technological process of forming tungsten metallization using chemical mechanical polishing. Dividing repeating structures into sepa-rate modules allows it to place all the necessary topology on a smaller number of photomasks, as well as to further sup-plement or reuse a set of photomasks for other test structures.

Keywords: test photomask, tungsten metallization, chemical mechanical polishing

Литература

1. С. И. Бабкин, С. И. Волков, К. С. Есенкин, А. С. Новоселов, А. А. Столяров. Многоуровневая металлизация для высокотемпературной микроэлектроники. Электронная техника. Серия 2. Полупроводниковые приборы. Выпуск 2 (241) 2016, с. 33-44

2. J. Chen, J-P. Colinge Tungsten metallization technology for high temperature silicon-on-insulator devices. Materials Science and Engineering, 1995, B29, pp. 18-20.

3. S.H. Li, R.O. Miller. Chemical mechanical polishing in silicon processing. Volume 63. 2000. pp.2-3

4. MIT/SEMATECH 931AZ Cu CMP Characterization Test Chip

Исследование характеристик КНИ МОП транзисторов A- и F-типов в расширенном диапазоне температур

А.С. Новоселов[1], С.В. Румянцев[2]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, anton_novoselov@srisa.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, sergey_rumyancev@srisa.ru

Аннотация. Исследованы динамические и статические характеристики КНИ МОП транзисторов, с помощью измерений кольцевых генераторов, составленных из инверторов на транзисторах A- и F-типов с длиной канала 0,35 мкм, в расширенном диапазоне температур от минус 60 до плюс 300°C. Показана работоспособность кольцевых генераторов в рассматриваемом диапазоне температур, задержка на инвертор не превысила 86 пс для A-транзисторов и 144 пс для F-транзисторов, токи утечки не превысили 10-13 микроампер. Показана возможность моделирования динамических и статических характеристик транзисторов вплоть до температуры в 300°C с использованием модели BSIMSOI с максимальной относительной ошибкой в 6-11%.

Ключевые слова: КНИ, высокая температура, BSIMSOI, модель, кольцевой генератор

1. Введение

Транзисторы, изготовленные по технологии «кремний на изоляторе» (КНИ), имеют преимущества по сравнению с традиционными транзисторами на «объемном» кремнии, такие как отсутствие тиристорного эффекта и меньшие токи утечки, из-за уменьшения площадей паразитных p-n переходов, поэтому технология КНИ, наряду с использованием материалов с широкой запрещенной зоной (нитрид галлия, карбид кремния), является основой для производства высокотемпературных микросхем.[1]

Для проектирования высокотемпературных микросхем необходимо иметь компактные модели КНИ транзисторов, которые корректно описывают характеристики транзисторов в расширенном диапазоне температур. В настоящий момент существует несколько компактных моделей КНИ МОП транзисторов: BSIMSOI, HISIM-SOI, EKV-SOI и др. В статье [2] был проведен анализ пригодности моделей BSIMSOI и HISIM-SOI для моделирования статических характеристик отдельных КНИ транзисторов, который показал, что обе модели корректно описывают температурные зависимости характеристик транзисторов, таких как пороговое напряжение и ток насыщения, однако анализа динамических характеристик произведено не было. Динамические характеристики КНИ транзисторов можно оценить с помощью измерений параметров кольцевого генератора, структуры, представляющей из себя цепочку из нечетного количества инверторов или других логических элементов.[3]

В данной работе были проведены измерения кольцевых генераторов, составленных из инверторов на КНИ транзисторах с затворами A- и F-типа[4], проведено SPICE-моделирование схем кольцевых генераторов на основе модели BSIM-SOI в диапазоне температур от минус 60 до плюс 300°C, произведено сравнение динамических характеристик модели и экспериментальных данных.

2. Описание образцов и методов исследования

Исследуемая модель транзисторов была получена на основе измерений вольт-амперных и вольт-фарадных характеристик транзисторов A и F-типов различных типоразмеров с минимальной длиной затвора L=0,35 мкм в диапазоне температур от минус 60 до плюс 300°C. Для оценки динамических параметров модели было выбрано 2 кольцевых генератора на A-(КГ_A) и F-транзисторах(КГ_F), построенных по одинаковой схеме (рис.1) из одного элемента И-НЕ, 20 инверторов, делителя частоты на 1024 и буфера. Инверторы составлены из транзисторов следующих размеров: КГ_A - Wn=2,1 мкм, Wp=3,1 мкм, Ln=Lp=0,35 мкм, КГ_F - Wn=1,5 мкм, Wp=2,75 мкм, Ln=Lp=0,35 мкм. Элемент И-НЕ запускает и останавливает генератор, тем самым позволяя измерить статический ток потребления кольцевого генератора. Моделирование схем кольцевых генераторов проводилось с учетом паразитных цепей, экстрагированных из топологии на SPICE-симуляторе Spectre Version 19.1 фирмы Cadence Design Systems. Измерение частоты генерации и статического тока потребления коль-

цевых генераторов проводилось на автоматизированном аппаратно-программном комплексе на основе оборудования фирмы Keysight Technologies (параметрический анализатор B1500A, матричный коммутатор E5250, осциллограф DSO6104L) и зондовой станции фирмы SUSS Microtech с термосистемой фирмы ATT, позволяющей проводить измерения в диапазоне от -60 до +300°C. Режимы измерения частоты (freq) и статического тока потребления(Ioff):

- Freq - Vdd=3,3 В, Ven=3,3 В, выход подключен к осциллографу.
- Ioff - Vdd=3,3 В, Ven=0,0 В

Частота кольцевого генератора затем пересчитывалась в задержку на один инвертор по следующей формуле (1):

$$Td = \frac{1}{2 \cdot N \cdot Freq \cdot 1024}, \qquad (1)$$

где N=21 – число инверторов в кольцевом генераторе, Freq – измеренная частота кольцевого генератора, 1024 – коэффициент делителя частоты.

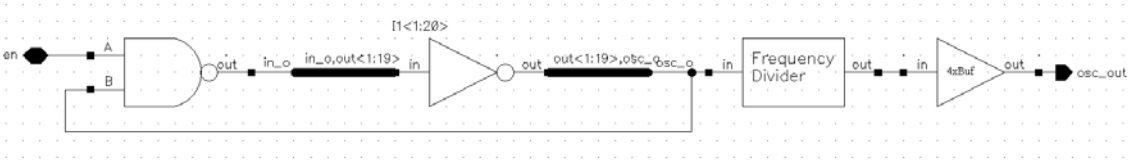


Рис. 1. Электрическая схема исследуемого кольцевого генератора

3. Основные результаты

На рисунке 2 представлены выходные сигналы кольцевого генератора A-типа снятые при температурах 25 и 300°C, видно, что кольцевой генератор сохраняет работоспособность при температуре 300°C при уменьшении частоты генерации с 437кГц до 284кГц. Кольцевой генератор на транзисторах F-типа также работоспособен при температуре 300°C, его частота снизилась с 285кГц до 173кГц.

Рис. 2. Результаты измерений частоты генерации кольцевого генератора на осциллографе

Изменение задержки на инвертор и статического тока потребления КГ A- и F-типов в зависимости от температуры представлено на рисунках 3 и 4 соответственно, статические токи F и A- генераторов получились примерно одинаковыми, поэтому, чтобы не загромождать рисунок представлена зависимость только для A-генератора. Сплошными линиями на рисунках указаны результаты, полученные при моделировании. Результаты измерений показали увеличение статического тока потребления генераторов на 5 порядков до единиц микроампер, что согласуется с результатами, полученными для одиночных транзисторов при проведении экстракции параметров модели.

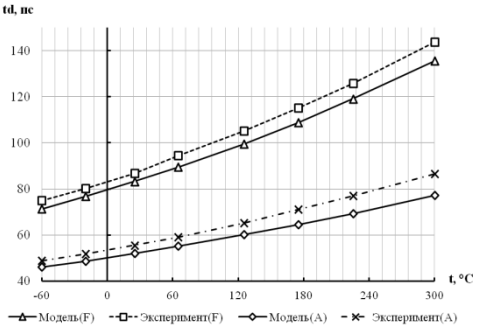


Рис. 3. Зависимость задержки на один инвертор кольцевых генераторов F- и A- от температуры

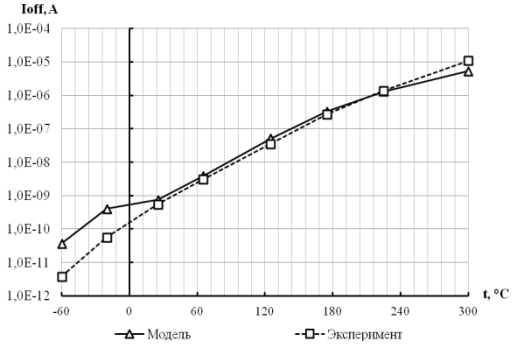


Рис. 4. Зависимость статического тока потребления кольцевого генератора от температуры

Задержка на инвертор увеличилась для A-генератора от 49 пс до 86 пс, при этом максимальная относительная ошибка моделирования составила 11 процентов, для F-генератора – от 75 пс до 144 пс, при этом максимальная относительная ошибка моделирования составила 6 процентов. Для оценки работоспособности модели необходимо произвести моделирование схемы при различных напряжениях питания. Результаты сравнения моделирования и данных эксперимента при температурах -60, 25 и 300°C и напряжениях питания 2,97 В, 3,3 В, 3,63 В для КГ F и A-типов представлены в таблицах 1 и 2.

Таблица 1. Сравнение результатов моделирования и эксперимента при разных напряжениях питания для КГ_F

КГ_F	**Задержка на инвертор, пс**					
	Vdd=2,97 В		**Vdd=3,3 В**		**Vdd=3,63В**	
Температура	**Модель**	**Эксп.**	**Модель**	**Эксп.**	**Модель**	**Эксп.**
-60	76	80	71	75	68	71
25	89	94	83	87	79	82
300	146	156	136	144	126	134

Таблица 2. Сравнение результатов моделирования и эксперимента при разных напряжениях питания для КГ_A

КГ_A	**Задержка на инвертор, пс**					
	Vdd=2,97 В		**Vdd=3,3 В**		**Vdd=3,63В**	
Температура	**Модель**	**Эксп.**	**Модель**	**Эксп.**	**Модель**	**Эксп.**
-60	49	52	46	49	44	46
25	56	60	52	55	49	52
300	83	94	77	86	72	80

Из представленных таблиц следует, что данные моделирования при изменении напряжения питания на ±10% соответствуют результатам эксперимента, при увеличении относительной ошибки с ростом температуры от 4% при минус 60°C до 11% при плюс 300°C для КГ_A, и от 3% при минус 60°C до 6% при плюс 300°C для КГ_F.

4. Заключение

Проведенные измерения показали работоспособность кольцевых генераторов составленных из КНИ транзисторов A- и F-типов в диапазоне температур от -60 до +300°C. Совпадение результатов измерений и моделирования в диапазоне напряжений питания 3,3±0,33В с максимальной ошибкой в 6-11% позволяет говорить о возможности использования модели BSIMSOI для моделирования высокотемпературных схем на основе технологии КНИ с проектными нормами 0,35 мкм.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН (выполнение фундаментальных научных исследований) по теме № 0065-2019-0018 «Исследование и построение моделей и конструкций элементов микроэлектроники в расширенном диапазоне температур (-60…300°C)».

Study of Characteristics of SOI MOS Transistors of A- and F-Types in an Extended Temperature Range

Novoselov A.S., Rumyancev S.V.

Abstract. The dynamic and static characteristics of SOI MOS transistors were investigated using measurements of ring oscillators composed of inverters on A- and F-type transistors with a channel length of 0.35 um, in an extended temperature range from minus 60 to plus 300°C. The efficiency of ring oscillators in the considered temperature range is shown, the delay per inverter did not exceed 86 ps for A-transistors and 144 ps for F-transistors, leakage currents did

not exceed 10-13 microamperes. The possibility of modeling the dynamic and static characteristics of transistors up to a temperature of 300°C using the BSIMSOI model with a maximum relative error of 6-11% is shown..

Keywords: SOI, high temperature, BSIMSOI, model, ring oscillator

Литература

1. J-P. Colinge Silicon-on-insulator technology: materials to VLSI. 3rd ed. Kluwer Academic Publishers; 2004.

2. Яшин Г.А., Амирханов А.В., Глушко А.А., Макарчук В.В., Новоселов А.С. Анализ пригодности современных компактных моделей КНИ МОП-транзисторов к моделированию в расширенном диапазоне температур. Труды НИИСИ РАН, 2018, т.8, №3, с.150-156.

3. M. Bhushan, M. B. Ketchen Microelectronic Test Structures for CMOS Technology, Springer, Science+Business Media, LLC 2011.

4. М.Г.Чистяков Исследование топологии полевых транзисторов для библиотеки стандартных функциональных узлов с минимальной проектной нормой 0,25 мкм. Сборник трудов 14 молодежной международной научно-технической конференции «Наукоемкие технологии и интеллектуальные системы 2012» - М.: МГТУ им. Н.Э. Баумана, 2012. – с. 231-237.

Компьютерное моделирование адиабатических и неадиабатических тепловых процессов в электронных системах при наличии и отсутствии вентиляционных отверстий на границе расчетной области

А.Г. Мадера[1], М.Ж. Акжолов[2], П.И. Кандалов[3]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, alexmadera@mail.ru; 8-903-733-1812

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, ak-1@mail.ru

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, petrki87@gmail.com

Аннотация. Приведены результаты численного моделирования конвективного теплообмена тепловыделяющих элементов с воздушной средой в электронных системах как при наличии, так и отсутствии вентиляционных отверстий на ограничивающих стенках.

Ключевые слова: компьютерное моделирование, электронные системы, тепловыделяющие элементы, конвективный теплообмен, расчетная область.

1. Введение

В данной работе проводится сравнительный анализ теплогазодинамических процессов вблизи тепловыделяющего элемента (ТВЭ) внутри расчетной области в условиях принудительной конвекции как при наличии, так и отсутствии выходных отверстий на ее границах, обеспечивающих естественный теплообмен нагреваемого ТВЭ воздуха с окружающей средой.

Как показано в работах [1,2], при тепловом проектировании электронных систем (ЭС) необходимо учитывать такие факторы, как:

– зависимость мощностей тепловыделения ТВЭ от температуры собственного разогрева, а температуры ТВЭ, в свою очередь, от изменившейся мощности (тепловая обратная связь);

– интервально стохастический характер тепловых процессов, вызванных интервальным характером определяющих тепловые процессы параметров [2,3,4];

– неоднородность и сложность формы множества ТВЭ, входящих в конструкцию ЭС [3, 4, 5, 6] и др.;

– сопряженный характер теплообмена [7, 8, 9].

В работах [10-17] приведены результаты математического и компьютерного моделирования ламинарно-турбулентного конвективного теплообмена ТВЭ с воздушной средой внутри ЭС. При моделировании принималось, что газовая среда идеальная, вязкость отсутствует, процессы теплопроводности в газе пренебрежимо малы по сравнению с конвективными процессами.

Данная работа является продолжением работы по изучению и анализу динамики тепловых процессов около ТВЭ при различных граничных условиях. В [18] изложены результаты компьютерного моделирования теплогазодинамических процессов при различных условиях (адиабатические и неадиабатические) на внешних границах. В [19, 20] изложены результаты компьютерного моделирования в условиях принудительного охлаждения ТВЭ ЭС.

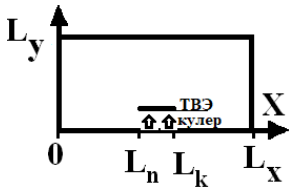


Рис. 1. Геометрия расчетной области

Схематически расчетная область показана на рисунке (рис.1). Размеры расчетной области составляли L_x = 0.18 м, L_y = 0.09 м, у основания области $y = 0$ в первой расчетной ячейке при $L_n \le x \le L_k$ находится ТВЭ. На нижней границе участка в интервале [L_k , L_n] задан источник тепла (ТВЭ) длиной 40 мм. Начальная температура воздуха T_0 = 300°К. Расчетная область разбивалась на 90x45 квадратных ячеек со сторонами $d_x = d_y$ = 0.002 м, шаг по времени $dt = 10^{-8}$ сек. Мощность источника тепла (ТВЭ) равна W=30 Вт. Граничные условия, соответствующие нагреванию в источнике тепла задаются в виде

[10]:

$$E(x_{гр}, y_{гр}, t) = \frac{Wt}{\rho(x_{гр}, y_{гр}, t)dxdy} \quad ,$$

где E – полная энергия, ρ – плотность газа.

Процесс охлаждение ТВЭ при обдуве моделируется следующим математическим соотношением:

$$W = W_0 e^{av} ,$$

где a = -0,312, v – скорость воздушного потока при обдуве кулером.

Математическая модель газодинамических процессов, протекающих при теплообмене ТВЭ со средой моделируется системой уравнений газовой динамики Эйлера и решается численно методом крупных частиц [17].

2. Начальные условия

В качестве начальных условий используются параметры невозмущенного потока, которые задаются во всех расчетных точках области:

$\rho^0_{i,j} = \rho^0_\infty = 1.225$ *кг/м*^{3},
$u^0_{i,j} = 0.0$,
$v^0_{i,j} = 0.0$,
$P^0_{i,j} = 1.01*10^5$ *Па*,
$E^0_{i,j} = E^0_\infty = P^0_{i,j} / [(\gamma\text{-}1)\ \rho^0_{i,j}]$,
$T^0_{i,j} = 300$ *K*,

где u,v – скорости по оси OX и OУ, P – давление, i=1,2,3,...,N1; j=1,2,3,...,M1 (рис. 2).

В расчетном поле воздух покоится, значения плотности, давления, температура и полной энергий постоянны.

2.1. Адиабатические условия на внешних границах

Адиабатические процессы моделировались следующими граничными условиями на внешних границах области (рис. 2):

– на левой границе AB

$\rho^n_{1,j} = \rho^n_{2,j}, u^n_{1,j} = - u^n_{2,j}$,
$v^n_{1,j} = v^n_{2,j}, E^n_{1,j} = E^n_{1,j}$,

– на нижней границе AD:

$\rho^n_{i,1} = \rho^n_{i,2}, u^n_{i,1} = u^n_{i,2}$,
$v^n_{i,1} = - v^n_{i,2}$, $E^n_{i,1} = E^n_{i,2}$,

– на верхней внешней границе BC расчетного поля

$\rho^n_{i,M1} = \rho^n_{i,M1-1}, u^n_{i,M1} = u^n_{i,M1-1}$,
$v^n_{i,M1} = -v^n_{i,M1-1}, E^n_{i,M1} = E^n_{i,M1-1}$,

– на правой внешней границе CD расчетного поля

$\rho^n_{N1,j} = \rho^n_{N1-1,j}, u^n_{N1,j} = -u^n_{N1-1,j}$,
$v^n_{N1,j} = v^n_{N1-1,j}, E^n_{N1,j} = E^0_\infty$.

Приведенные граничные условия обеспечивают отсутствие перетекания массы через внешние границы (твердая стенка), равно как и энергии, импульса и тепла.

Поток массы через правую границу CD определяется следующими соотношениями:

$$\Delta M^n_{(N1-1)/2,j} = \rho^n_{N1-1,j} \frac{u^n_{N1-1,j} + u^n_{N1,j}}{2} \Delta t \Delta y \ ,$$

если $u^n_{N1-1,j} + u^n_{N1,j} \geq 0$,

$$\Delta M^n_{(N1-1)/2,j} = \rho^n_{N1,j} \frac{u^n_{N1-1,j} + u^n_{N1,j}}{2} \Delta t \Delta y \ ,$$

если $u^n_{N1-1,j} + u^n_{N1,j} < 0$.

Потоки массы на левой, нижней и верхних границах определялись аналогичными образами. Отметим, что потоки массы несут с собой потоки энергии, импульса и тепла во внешнюю среду.

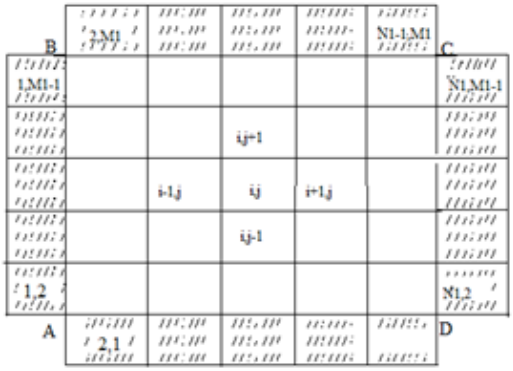


Рис. 2. Расчетная область

2.2. Граничные условия при наличии отверстий на границе расчетной области

Отверстия в верхней части правой и левой стенках моделировались следующими граничными условиями (рис. 2):

– на левой внешней границе AB

$\rho^n_{1,j} = \rho^n_{2,j}$, $u^n_{1,j} = u^n_{2,j}$,
$v^n_{1,j} = v^n_{2,j}$, $E^n_{1,j} = E^n_{1,j}$,

– на правой внешней границе CD расчетного поля

$\rho^n_{N1,j} = \rho^n_{N1-1,j}$, $u^n_{N1,j} = u^n_{N1-1,j}$,
$v^n_{N1,j} = v^n_{N1-1,j}$, $E^n_{N1,j} = E^n_\infty$.

где индекс j меняется от 38 по 42, т.е. используется по пять ячеек для моделирования каждого отверстия и через данные отверстия обеспечивается свободное перетекание потоков массы импульса и энергии.

3. Результаты и анализ

I. Для момента времени t=0.05 сек. от начала расчета (рис. 3а, 3б), показаны температурные поля и поля векторов скоростей соответственно для закрытых стен (условия адиабатические), а на рис. 3в и 3г показаны температурные поля и поля векторов скоростей соответственно при

наличии отверстий на горнице расчетной области. Скорость воздушного потока от кулера составляет V=1 м/с.

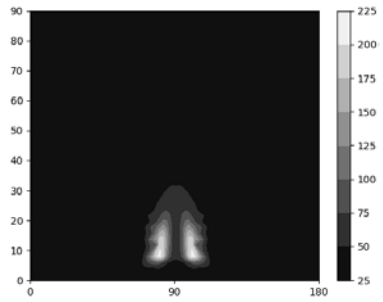


Рис.3а. Температурное поля воздуха в момент времени t=0,05 сек со скоростью обдува V=1 м/с при адиабатических условиях.

Рис.3б. Поле вектор скоростей воздуха при t=0,05 сек со скоростью обдува V=1 м/с. Закрытыми внешними стенками.

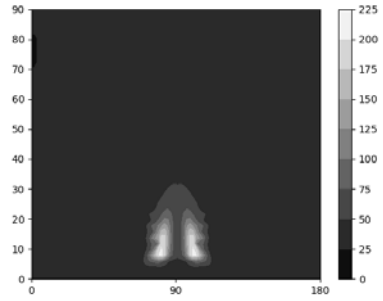


Рис.3в. Температурное поля воздуха в момент времени t=0,05 сек со скоростью обдува V=1 м/с при наличии отверстий на границах расчетной области.

Рис.3г. Поле вектор скоростей при t=0,05 сек со скоростью обдува V=1 м/с при наличии отверстий на границах расчетной области.

II. Для момента времени t=0.10 сек. от начала расчета (рис. 4а, 4б), показаны температурные поля и поля векторов скоростей соответственно для закрытых стен (условия адиабатические), а на рис. 4в и 4г показаны температурные поля и поля векторов скоростей соответственно при наличии отверстий на горнице расчетной области. Скорость воздушного потока от кулера составляет V=1 м/с.

III. Для момента времени t=1.00 сек. от начала расчета (рис. 5а, 5б), показаны температурные поля и поля векторов скоростей соответственно для закрытых стен (условия адиабатические), а на рис. 5в и 5г показаны температурные поля и поля векторов скоростей соответственно при наличии отверстий на горнице расчетной области. Скорость воздушного потока от кулера составляет V=1 м/с.

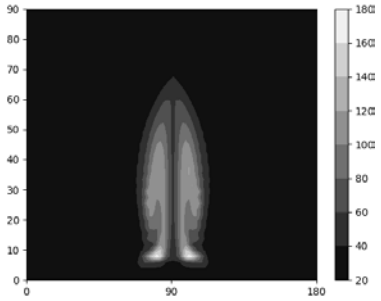


Рис.4а Температурное поля воздуха в момент времени t=0,10 сек со скоростью обдува V=1 м/с при адиабатических условиях.

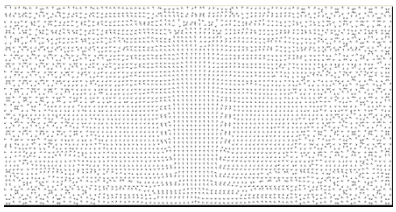


Рис.4б. Поле вектор скоростей воздуха при t=0,10 сек со скоростью обдува V=1 м/с. Закрытыми внешними стенками.

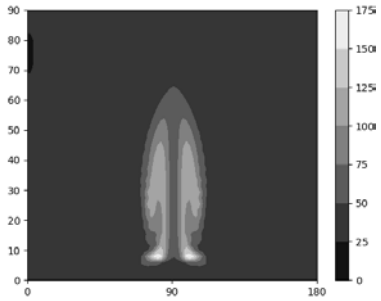


Рис.4в. Температурное поля воздуха в момент времени t=0,10 сек со скоростью обдува V=1 м/с при наличии отверстий на границах расчетной области.

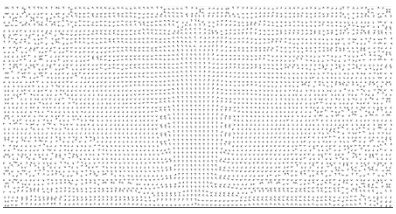


Рис.4г. Поле вектор скоростей воздуха при t=0,10 сек со скоростью обдува V=1 м/с при наличии отверстий на границах расчетной области.

IV. Для момента времени t=2.00 сек. от начала расчета (рис. 6а, 6б), показаны температурные поля и поля векторов скоростей соответственно для закрытых стен (условия адиабатические), а на рис. 6в и 6г показаны температурные поля и поля векторов скоростей соответственно при наличии отверстий на горнице расчетной области. Скорость воздушного потока от кулера составляет V=1 м/с.

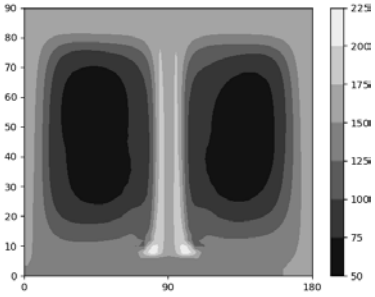


Рис.5а. Температурное поля воздуха в момент времени t=1,00 сек со скоростью обдува V=1 м/с при адиабатических условиях.

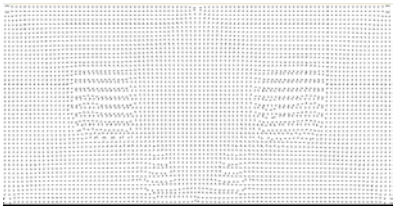


Рис.5б. Поле вектор скоростей воздуха в момент времени t=1,00 сек со скоростью обдува V=1 м/с. Закрытыми внешними стенками.

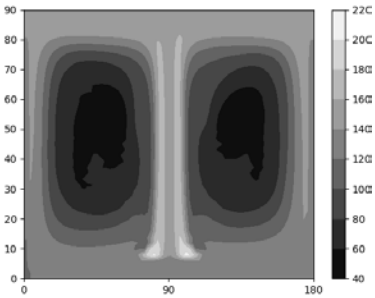


Рис.5в. Температурное поля воздуха в момент времени t=1,00 сек со скоростью обдува V=1 м/с при наличии отверстий на границах расчетной области.

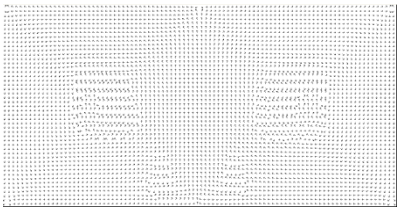


Рис.5г. Поле вектор скоростей воздуха в момент времени t=1,00 сек со скоростью обдува V=1 м/с при наличии отверстий на границах расчетной области.

V. Для момента времени t=0.05 сек. от начала расчета (рис. 7а, 7б), показаны температурные поля и поля векторов скоростей соответственно для закрытых стен (условия адиабатические), а на рис. 7в и 7г показаны температурные поля и поля векторов скоростей соответственно при наличии отверстий на горнице расчетной области. Скорость воздушного потока от кулера составляет V=3 м/с.

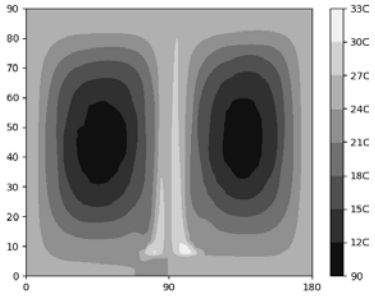


Рис.6а. Температурное поля воздуха в момент времени t=2,00 сек со скоростью обдува V=1 м/с при адиабатических условиях.

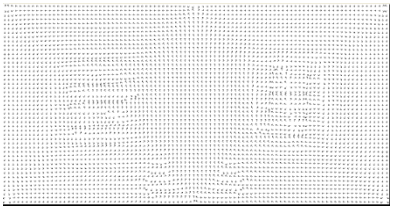


Рис.6б. Поле вектор скоростей воздуха в момент времени t=2,00 сек со скоростью обдува V=1 м/с. Закрытыми внешними стенками.

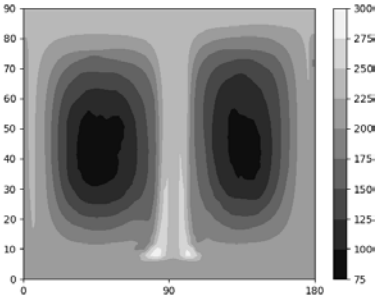


Рис.6в. Температурное поля воздуха в момент времени t=2,00 сек со скоростью обдува V=1 м/с при наличии отверстий на границах расчетной области.

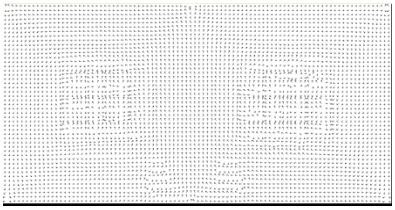


Рис.6г. Поле вектор скоростей воздуха в момент времени t=2,00 сек со скоростью обдува V=1 м/с при наличии отверстий на границах расчетной области.

VI. Для момента времени t=0.10 сек. от начала расчета (рис. 8а, 8б), показаны температурные поля и поля векторов скоростей соответственно для закрытых стен (условия адиабатические), а на рис. 8в и 8г показаны температурные поля и поля векторов скоростей соответственно при наличии отверстий на горнице расчетной области. Скорость воздушного потока от кулера составляет V=3 м/с.

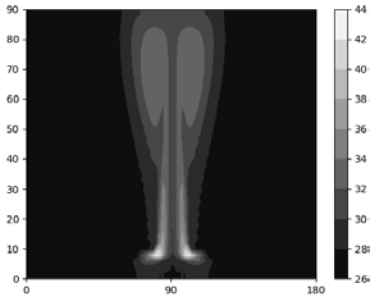


Рис.7а. Температурное поля воздуха в момент времени t=0,05 сек со скоростью обдува V=3 м/с при адиабатических условиях.

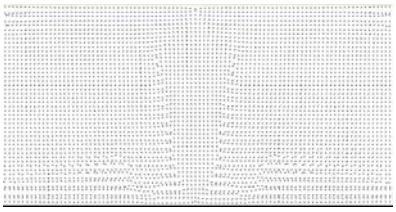


Рис.7б. Поле вектор скоростей воздуха в момент времени t=0,05 сек со скоростью обдува V=3 м/с. Закрытыми внешними стенками.

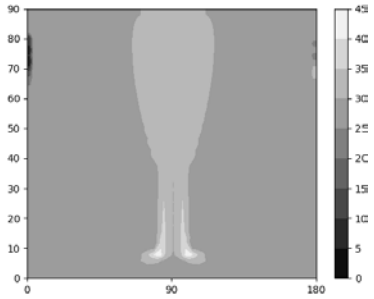


Рис.7в. Температурное поля воздуха в момент времени t=0,05 сек со скоростью обдува V=3 м/с при наличии отверстий на границах расчетной области.

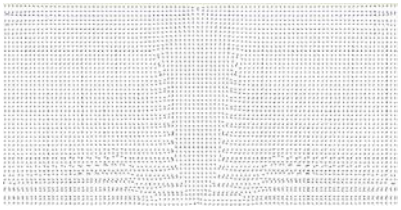


Рис.7г. Поле вектор скоростей воздуха в момент времени t=0,05 сек со скоростью обдува V=3 м/с при наличии отверстий на границах расчетной области.

VII. Для момента времени t=1.00 сек. от начала расчета (рис. 9а, 9б), показаны температурные поля и поля векторов скоростей соответственно для закрытых стен (условия адиабатические), а на рис. 9в и 9г показаны температурные поля и поля векторов скоростей соответственно при наличии отверстий на горнице расчетной области. Скорость воздушного потока от кулера составляет V=3 м/с.

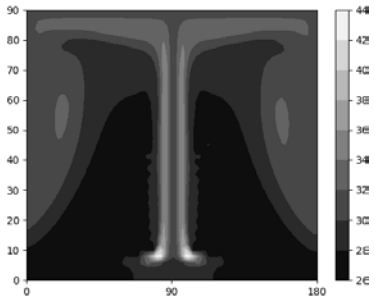


Рис.8а. Температурное поля воздуха в момент времени t=0,10 сек со скоростью обдува V=3 м/с при адиабатических условиях.

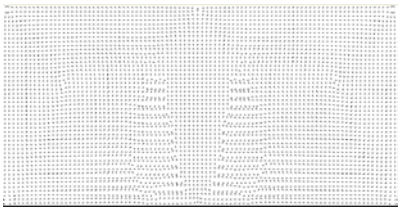


Рис.8б. Поле вектор скоростей воздуха в момент времени t=0,10 сек со скоростью обдува V=3 м/с. Закрытыми внешними стенками.

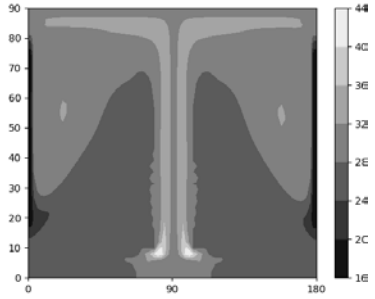


Рис.8в. Температурное поля воздуха в момент времени t=0,10 сек со скоростью обдува V=3 м/с при наличии отверстий на границах расчетной области.

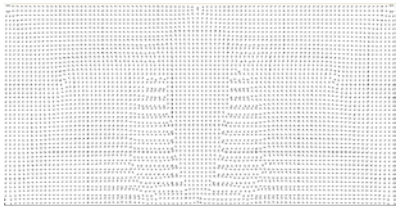


Рис.8г. Поле вектор скоростей воздуха в момент времени t=0,10 сек со скоростью обдува V=3 м/с при наличии отверстий на границах расчетной области.

VIII. Для момента времени t=2.00 сек. от начала расчета (рис. 10а, 10б), показаны температурные поля и поля векторов скоростей соответственно для закрытых стен (условия адиабатические), а на рис. 10в и 10г показаны температурные поля и поля векторов скоростей соответственно при наличии отверстий на горнице расчетной области. Скорость воздушного потока от кулера составляет V=3 м/с.

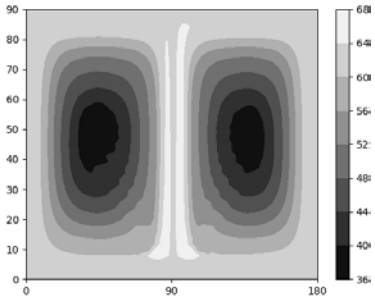


Рис.9а. Температурное поля воздуха в момент времени t=1,00 сек со скоростью обдува V=3 м/с при адиабатических условиях.

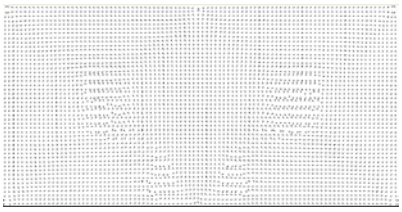


Рис.9б. Поле вектор скоростей воздуха в момент времени t=1,00 сек со скоростью обдува V=3 м/с. Закрытыми внешними стенками.

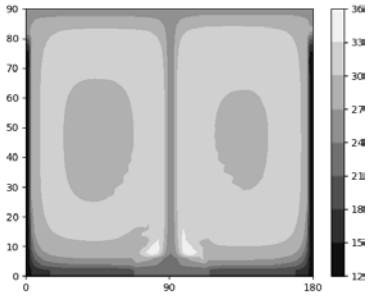


Рис.9в. Температурное поля воздуха в момент времени t=1,00 сек со скоростью обдува V=3 м/с при наличии отверстий на границах расчетной области.

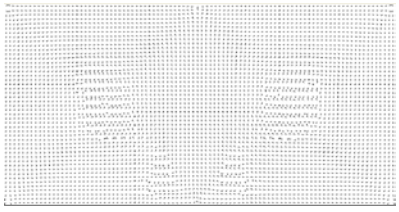


Рис.9г. Поле вектор скоростей воздуха в момент времени t=1,00 сек со скоростью обдува V=3 м/с при наличии отверстий на границах расчетной области.

На рисунках, где показаны температурные поля по вертикальной оси, отложены значения температуры в ^{o}C, по горизонтальной оси значения координат относительно расчетной области. Справа приведена черно-белая палитра с оттенками серого, идентифицирующая значения температур в различных участках процесса и области.

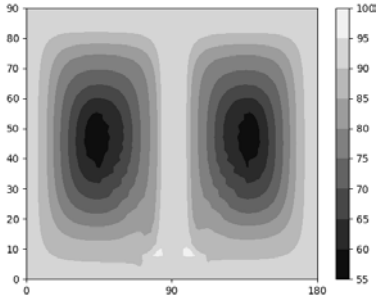


Рис.10а. Температурное поля воздуха в момент времени t=0,10 сек со скоростью обдува V=3 м/с при адиабатических условиях.

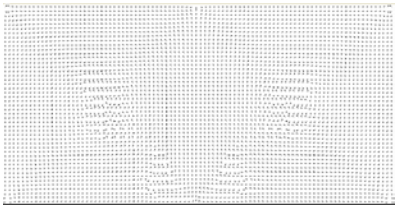


Рис.10б. Поле вектор скоростей воздуха в момент времени t=2,00 сек со скоростью обдува V=3 м/с. Закрытыми внешними стенками.

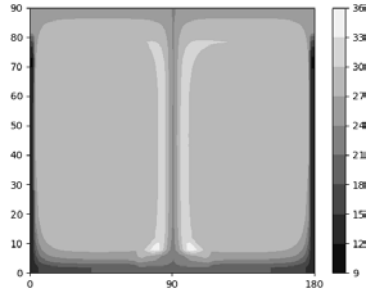


Рис.10в. Температурное поля воздуха в момент времени t=2,00 сек со скоростью обдува V=3 м/с при наличии отверстий на границах расчетной области.

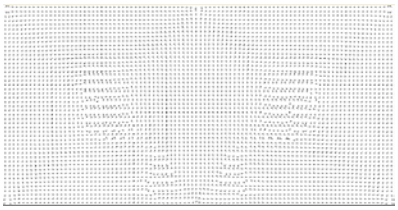


Рис.10г. Поле вектор скоростей воздуха в момент времени t=2,00 сек со скоростью обдува V=3 м/с при наличии отверстий на границах расчетной области.

Анализ полученных результатов показывает следующее. При нагреве области, занятой ТВЭ, над ним развивается конвективный процесс, происходящий под действием давления и сил гравитации (максимальная скорость $\leq \sqrt{2g_0H}$, где H =0.09 м – высота расчетной области). Полученное изображение развития конвективных процессов в воздухе над областью с ТВЭ достаточно хорошо отражает моделируемое физическое явление. Тепловые потоки, возникшие за счет нагревания ТВЭ развиваются плавно (монотонно) начиная с приповерхностного слоя, а дальнейшее развитие процесса во времени показывает расширение области конвекции, охватывающей все больший объем воздуха. По результатам моделирования хорошо видно, что в один и тот же момент времени размер охватываемой тепловым возмущением области изменяется в зависимости от граничных условий с закрытыми внешними стенками и открытыми окошками на правых и левых боковых стенках. При наличии окошек на боковых стенках происходит активное перемешивание нагреваемого источников воздуха с окружающей средой, за счет чего фронт охвата зоны теплового возмущения шире, по сравнению с закрытыми стенками (теплоизолированной системой), а эффект охлаждения заметно больше по сравнению с закрытой областью, что достаточно убедительно демонстрируется на приведенных рисунках.

4. Заключение

По результатам моделирования хорошо видно, что в один и тот же момент времени размеры и формы, охватываемой тепловым возмущением области, изменяются по-разному в зависимости от внешних граничных условии, т.е. адиабатические и неадиабатические процессы.

Полученное изображение развития конвективных процессов в воздухе над ТВЭ достаточно хорошо отражает моделируемое физическое явление. Тепловые потоки, возникшие за счет нагревания ТВЭ развиваются плавно (монотонно) начиная с приповерхностного слоя, а дальнейшее развитие процесса во времени показывает расширение области охвата конвекцией все большего объема воздуха.

На основе выше изложенной математической модели можно провести численные эксперименты для изучения газодинамических и тепловых процессов на ТВЭ и вблизи него.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН (Фундаментальные исследования 47 ГП) по теме № 0065-2019-0001 "Математическое обеспечение и инструментальные средства для моделирования, проектирования и разработки элементов сложных технических систем, программных комплексов и телекоммуникационных сетей в различных проблемно-ориентированных областях" (АААА-А19-119011790077-1).

Computer Modeling of Adiabatic and Nonadiabatic Thermal Processes in Electronic Systems at the Presence and Absence of Ventilation Holes at the Boundary on the Computational Domain

A.G. Madera, M.J. Akjolov, P.I. Kandalov

Abstract. The results of numerical modeling of convective heat exchange of heat arises elements with an air medium in electronic systems are presented, both at the presence and absence of ventilation holes at the boundary on the computational domain.

Keywords: computer modeling, electronic systems, heat arises elements, convective heat transfer, computational domain

Литература

1. А.Г. Мадера, М.Ж. Акжолов, И.Г. Лебо Моделирование развития процессов конвекции плюс теплопроводность в воздухе вблизи процессора. // Труды НИИСИ РАН, 2013, Т. 3, №1. С. 90 – 93.

2. А.Г. Мадера Концепция математического и компьютерного моделирования тепловых процессов в электронных системах // Программные продукты и системы. 2015. № 4. С.79 – 86

3. А.Г. Мадера Моделирование интервально стохастических нестационарных и нелинейных тепловых процессов в электронных системах // Труды НИИСИ РАН. 2015. Т. 5. №1. С. 5 – 9

4. П.И. Кандалов, А.Г. Мадера Моделирование температурных полей в многослойных структурах // Программные продукты и системы. 2008. № 4. С. 11

5. А.Г. Мадера Моделирование теплообмена в технических системах. – М.: НФ им. ак. В.А. Мельникова, 2005.

6. А.Г. Мадера, П.И. Кандалов Анализ интервально стохастических температурных полей технических систем // Программные продукты и системы. 2014. №4. С. 41-45.

7. А.Г. Мадера, П.И. Кандалов Моделирование температурных полей технических систем в условиях интервальной неопределенности // Тепловые процессы в технике. 2014.Т.6.№5.С.225-229.

8. А.Г. Мадера Математическое моделирование свободного конвективного теплообмена в электронных системах // Труды НИИСИ РАН, 2011, Том 1 №1. С. 31 – 37.

9. A.G. Madera Heat transfer from an extended surface at a stochastic heat-transfer coefficient and stochastic environmental temperature // International Journal of Engineering Science. 1996. Т. 34. № 9. С. 1093 – 1099.

10. М.Ж. Акжолов Численное моделирование конвективного теплообмена в электронной системе при различных мощностях источника тепла // Труды НИИСИ РАН, 2016, Т.6 №1. С. 62-63.

11. М.Ж. Акжолов Математическое моделирование конвективного теплообмена тепловыделяющего элемента с воздушной средой в электронной системе. // Труды НИИСИ РАН, 2015, Том 5, №2. С. 103 – 105.

12. М.Ж. Акжолов, П.И. Кандалов, И.Г. Лебо, А.Г. Мадера Компьютерное моделирование конвективных процессов в воздушной среде вблизи электронных устройств. // Труды НИИСИ РАН, 2011, Том. 1, №2. С. 44 – 46.

13. М.Ж. Акжолов Влияние мощностей потребления и температуры окружающей среды на интегрально-стохастические тепловые процессы в электронном модуле. (статья) // Труды НИИСИ РАН. 2017, Том 7, №4. С.51-56.

14. М.Ж. Акжолов Анализ конвективного теплообмена с воздушной средой в электронных системах методом крупных частиц. Седьмая Российская национальная конференция по теплообмену (РНКТ-7): том 1. (22—26 октября 2018 г., Москва). С. 57-61.

15. М.Ж. Акжолов Воздействие температуры окружающей среды и мощностей потребления на тепловые процессы электронных систем в условиях стохастической неопределенности. Седьмая Российская национальная конференция по теплообмену (РНКТ-7): Том 3. (22—26 октября 2018 г., Москва). С. 93-95.

16. М.Ж. Акжолов Численное исследование температурного поля воздуха при нагревании от теплового источника с учетом и без учета теплопроводности. // Труды НИИСИ РАН. 2019, Том 9, № 2. С. 46-49.

17. О.М. Белоцерковский, Ю.М. Давыдов Метод крупных частиц в газовой динамике. – М.: Наука, 1982.

18. А.Г. Мадера, М.Ж. Акжолов Исследование адиабатических и неадиабатических тепловых в электронных системах около тепловыделяющего элемента. // Труды НИИСИ РАН. 2020, Том 10, № 2, С. 18-22.

19. М.Ж. Акжолов, П.И. Кандалов Численное моделирование теплообмена тепловыделяющего элемента в окружающей среде в электронной системе при обдуве кулером. // Труды НИИСИ РАН. 2020, Том 10, № 2, С. 23-27.

20. М.Ж. Акжолов численное исследование зависимости конвективного охлаждения тепловыделяющего элемента электронной системы от скорости воздуха при обдуве кулером. // Труды НИИСИ РАН. 2020, Том 10, № 2. С. 28-33.

Аномальная теплопроводность в кремниевых цилиндрических наноструктурах

Н.В. Масальский[1]

[1]ФГУ ФНЦ НИИСИ РАН, г. Москва, volkov@niisi.ras.ru

Аннотация. Обсуждается решения уравнения, описывающего аномальное распространение тепла в кремниевых волокнах. Рассмотрены два случая начального теплового возмущения: узко параболической формы и формы близкой к пилообразной. Показано, что, решение затухает медленнее вблизи волнового фронта, наблюдаются ярко выраженные максимумы, которые могут быть обнаружены экспериментально. Полученные решения демонстрируют волновую природу и степенное затухание, что отличает их от решений классического уравнения теплопроводности. Разработана математическая модель температурных волн применительно к кремниевым наноразмерным структурам с цилиндрической геометрией на основе гиперболического уравнения теплопроводности. Этот подход позволяет эффективно определить тепловые свойства наноструктуры и может быть востребованным для того, чтобы выделить наиболее перспективные свойства наносистем.

Ключевые слова: аномальная теплопроводность, число Кнудсена, уравнение МКВ, тепловая волна

1. Введение

Новые аспекты в процессе теплопередачи возникли в связи с изменением отношения между «средним свободным пробегом» теплоносителей (l_{ph}) и соответствующим характерным размером низко-размерной системы L, которое выражено числом Кнудсена $Kn = l_{ph}/L$ [1, 2]. Вопреки классическим канонам, теплопроводность k уже зависит от длины решетки L_{lt}. Она теперь изменяется со степенной зависимостью поведения в виде $k \sim L_{lt}^{-\eta}$, при этом значение η находится в интервале 1/3…2/5 [2, 3]. Закон Фурье (а также классическая гидродинамика) справедлив в пределе очень малых чисел Кнудсена, то есть когда $Kn << 1$ [4, 5]. Действительно, число Кнудсена может увеличиваться из-за уменьшения L, например, при масштабировании транзисторных структур. Огромное количество экспериментальных данных по низкоразмерным системам выявило необходимость использования более сложных моделей теплопроводности, учитывающих и аномальную природу из-за конечной скорости распространения тепловых возмущений [1-3].

Аномальная природа тепловых процессов в низко размерных структурах преодолевает тепловой парадокс Фурье, который предсказывает бесконечную скорость распространения теплового сигнала, что парадоксально с физической точки зрения. Альтернатива закону Фурье – гиперболическое уравнение теплопроводности Максвелла-Каттанео-Вернотти (МКВ) [6, 7]. Оно позволяет изучать процессы на микроуровне, когда характерные длины пропорциональны нескольким длинам межатомных связей. Трансформация закона Фурье приводит к уравнению гиперболической теплопроводности МКВ, которое можно записать в виде [1].

$$(\partial_t^2 + \frac{1}{\tau}\partial_t)T = \frac{\beta}{\tau}\nabla^2 T \qquad (1)$$

где τ - время релаксации, β - коэффициент температуропроводности.

Следует отметить, что и такой подход не лишен противоречий. Например, возникают сложности в применении его к описанию тепловых процессов в транзисторных структурах, так как не удается однозначно определить время релаксации [3].

С точки зрения задач современной микроэлектроники - внедрение наноразмерных цилиндрических транзисторных архитектур напрямую связано с успешным решением задачи – повышения степени интеграции и быстродействия электронных микросхем [8]. Другим актуальным направлением применения кремниевых цилиндрических наноструктур является разработка масштабируемой измерительной платформы для определения физико-химических параметров наноразмерных биологических объектов [9]. В ближайшей перспективе решение проблем управления переносом тепла для изоляции или охлаждения, или для тонкого фононного управления в тепловых системах, где применение кремниевых нановолокон на ряду с углеродными нанотрубками, полосами графена и пористыми

материалами могут быть решающими [10]. Поэтому в настоящее время стимулируются исследования, направленные на изучение механизмов теплопередачи кремниевых цилиндрических наноструктур, разработки математических моделей и программ моделирования.

Цель работы численно изучить распространение тепловых импульсов различной формы и исследовать затухание и асимптотику теплового фронта применительно к кремниевым наноструктурам с цилиндрической геометрией.

2. Тестовые решения уравнения МКВ

Мы получим численно решения уравнения (1) для двух частных случаев форм начальных тепловых возмущений: узкого параболического импульса и импульса в форме прямоугольного треугольника с выпуклой гипотенузой. Их вид приведен на рисунке 1.

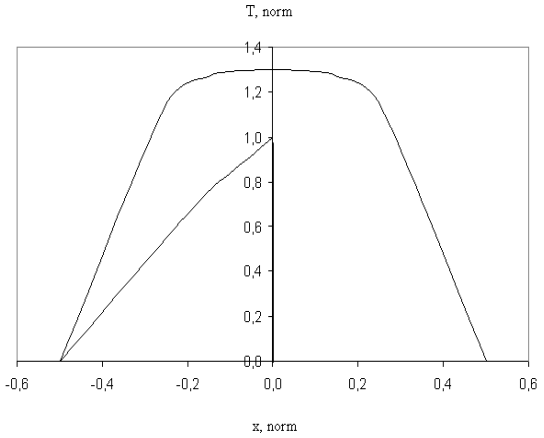


Рис. 1. Формы исходных тепловых сигналов, где 1 – параболический, 2- треугольный

Эти расчеты выполнены для иллюстрации ключевых свойств теплового фронта и анализа отличий его от закона Фурье. На рисунке 2 приведены численно рассчитанные распределения температуры в изотропной среде для параболического импульса для разных масштабов времени относительно времени релаксации τ .

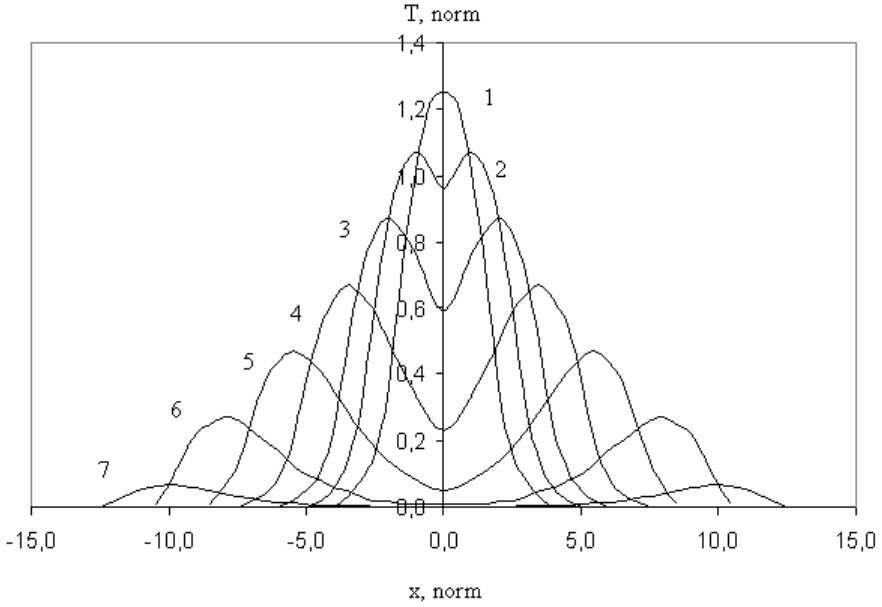


Рис. 2. Температурные распределения при разных t / τ , 1-2, 2-2.5, 3-3, 4-3.5, 5-4, 6-6 , 7-10

В случае МКВ температура резко понижается вдоль линии x= 0. При этом формируются две расходящиеся симметричные тепловые волны. В отличие от классического уравнения теплопроводности, решение уравнения (1) имеет четко выраженный волновой фронт. Затухание решения вблизи волнового фронта и нуля существенно различаются. Решение затухает медленнее вблизи волнового фронта, в результате чего наблюдаются ярко выраженные максимумы.

На рисунке 3 представлены расстояния между максимами волн.

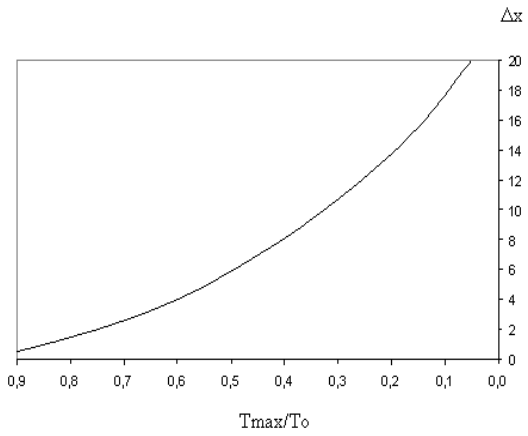


Рис. 3.

В масштабе температуры траектория волны является криволинейной. В пространстве она зависит от полуширины импульса воздействия и скорости звука в среде.

Напротив, по закону Фурье в данном случае наблюдается единственный максимум в точке x=0, который со временем затухает экспоненциально, не отклоняясь от оси x=0, что хорошо известно.

Затухание тепловых волн во временном масштабе по оси x=0 и линиям, проходящим через максимумы волн, существенно различаются, что иллюстрируется рисунком 4. При небольших временах, когда происходит формирование волн, эти две зависимости ведут себя одинаково. Далее проявляются существенные различия. Затухание вдоль центральной оси аномально увеличивается (значение температуры резко уменьшается) и примерно за один временной интервал относительная температура стремиться к нулю. Максимумы ведут себя более плавно постепенно снижаясь за каждый временной отрезок.

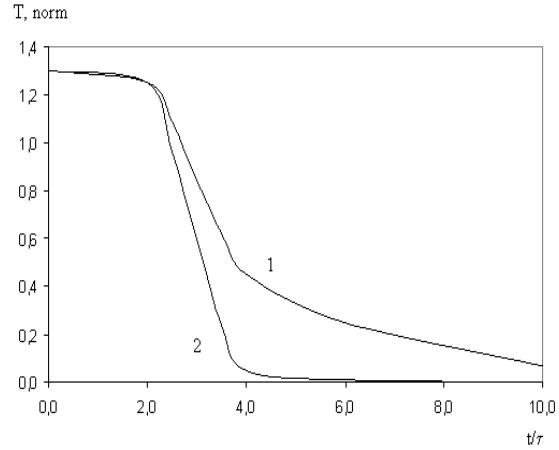


Рис. 4. Затухание волнового фронта, где 1 – максимум тепловой волны, 2 – вдоль оси x=0

Примерно также происходит процесс теплопередачи и при начальном треугольном воздействии, механизм формирования волн другой. На рисунке 5 приведены численно рассчитанные распределения температуры в изотропной среде для импульса 2 на рисунке 1 для разных масштабов времени t/τ.

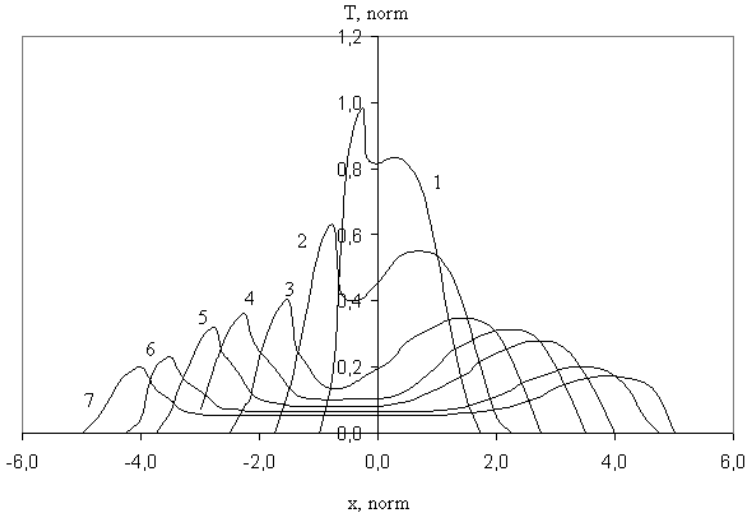


Рис. 5. Температурные распределения при разных t/τ, 1-0.2 2-0.5, 3-1.0, 4-1.5, 5-2.0, 6-2.5 , 7-3.0

Тепловая волна, распространяющаяся в отрицательном направлении, имеет ярко выраженный пик, который со временем размывается, но асимметрия волновой формы сохраняется. Напротив, тепловая волна, распространяющаяся в положительном направлении, имеет сглаженную вершину. и гладкое поведение в точке максимума.

Затухание тепловых волн во временном масштабе по оси x=0 и линиям, проходящим через

максимумы волн, существенно различаются, что иллюстрируется рис. 6. Во временном интервале 0…1 эти две зависимости ведут себя одинаково. Далее проявляются существенные различия. Затухание вдоль центральной оси продолжает увеличиваться примерно по гиперболическому закону. Затухание максимумов замедляется, что связано с перестройкой пространственного расположения (пространственной частоты). В начальном временном интервале «провал» между волнами не совпадает с центральной осью. Даже удаляется от нее. Затем он снова приближается к ней, что соответствует пологому отрезку на кривой 1. Далее максимумы плавно постепенно снижаются.

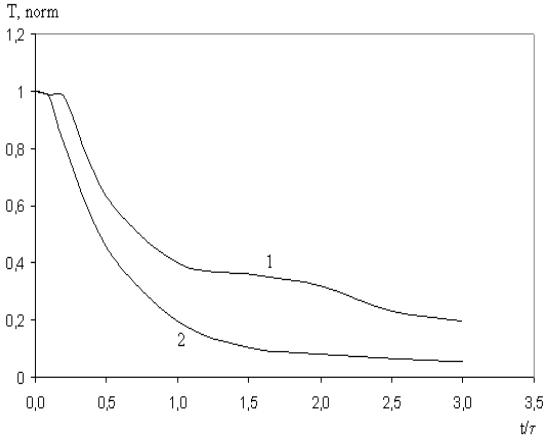


Рис. 6. Затухание волнового фронта, где 1 – максимум тепловой волны, 2 – вдоль оси x=0

На рисунке 7 представлены расстояния между максимумами волн. В данном случае траектория волны более плавная и близка к линейной.

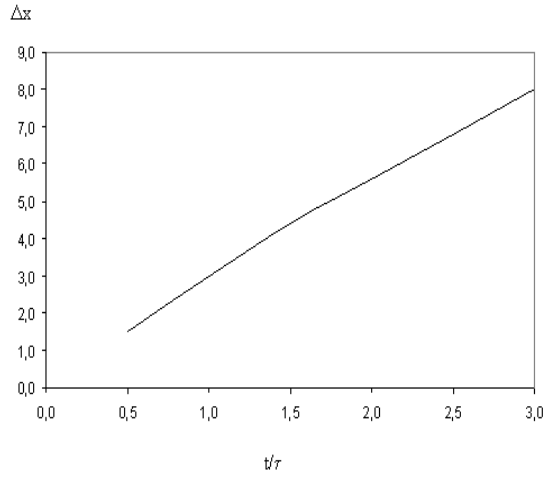


Рис. 7.

Полученные результаты являются исходными данными для анализа аномального распространения тепла в более сложных объектах, например, двумерных и трехмерных кристаллических структурах. Понимание процессов аномального распространения тепла особенно важно для анализа экспериментальных данных, которые вступают в противоречие с законом Фурье.

3. Температурные волны вдоль кремниевых цилиндрических структур

При изучении аномальной зависимости теплопроводности от характерного размера сложной наносистемы мы будем рассматривать цилиндрическую кремниевую (или любую полупроводниковую – это неважно) с круглым сечением наноструктуру. И самое главное – гладкостенную – это важно! т. е. при отсутствии обратного рассеяния. Основные параметры R-радиус поперечного сечения, который составляет двадцать десятка нанометров, L-продольная длина, которая больше диаметра устройства. Схематически это конструкция приведена на рисунке 8.

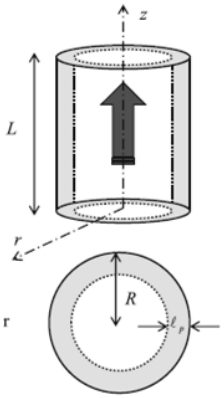


Рис. 8. Цилиндрическая наноструктура с круглым поперечным сечением.
Направление теплового потока показано стрелкой

В обобщенном случае с учетом теплообмена через боковые стенки наноструктуры уравнение (1) преобразуется в следующее уравнение [11]

$$\ddot{T}_t + \left(\frac{1}{\tau_R} + \frac{1}{\tau_{R\_exc}}\right)\dot{T}_t = U_0 \nabla^2 T - \left(\frac{T - T_{amb}}{\tau_R \tau_{R\_exc}}\right)$$

, (2)

где τ_R - время между столкновениями фононов, T_{amb} - температура окружающей среды, которая считается постоянной и однородной, $\tau_{R\,exc} = \frac{Rc_V}{2\sigma_{exc}}$ - время релаксации, обусловленное продольной дисперсией тепла, которое выражает запаздывание во времени, обусловленное только теплообменом на границе структуры, σ_{exc} - коэффициент теплообмена. Время $\tau_{R\,exc}$ не связано с процессами столкновений между

фононами, которые включены во время релаксации τ_{R}, $U_0 = (\frac{k_0}{\tau_R c_V})^{1/2}$ - коэффициент нормировки, c_{V} - удельная теплоемкость на единицу объема (в расчетах использовалось выражение Дебая).

Из уравнения (2) можно видеть, что из-за малой толщины проводника продольная дисперсия вводит конкурентную временную шкалу, которая влияет на эволюцию температуры. В пределах большого радиуса ($\mathrm{R} \to \infty$), или при идеальной изоляции $\tau_{\mathrm{R\,exc}} \to \infty$, (2) сводится к классическому телеграфному уравнению, возникающему в теории МКВ [5-7] .

Для простоты будем считать, что тепловые потери через боковые стенки незначительны. Тем не менее, из-за нескольких столкновений фононов со стенками, вклад скользящего (пристеночного) потока q_w, необходимо учитывать дополнительно к объемному тепловому потоку $\mathrm{q_b}$. Тогда общий тепловой поток Q запишем так: $\mathrm{Q} = \mathrm{q_b}(r) + q_w$

В стационарном случае вклад объемного теплового потока имеет следующий профиль в каждом поперечном сечении [12]

$$\mathrm{q_b}(r) = \frac{k}{4l_{ph}^{\,2}}(R^2 - r^2)\nabla T\,, \qquad (3)$$

где *r*-радиальное расстояние от оси z , $\nabla T = -T/L$ температурный градиент.

В принципе, вклад потока q_w ограничен тонкой областью вблизи стенок – слоем Кнудсена. Однако в наносистеме, характерная размерность которой сравнима с (или меньше) l_{ph}, слой Кнудсена покрывает всю область, а q_w можно принять за однородный вклад в общий тепловой поток. В случае гладких стенок вклад пристеночного теплового потока можно записать в виде [12]

$$\mathrm{q_w} = C\frac{kR}{2l_{ph}}\nabla T\,, \qquad (4)$$

Следует отметить, что уравнение (4) определяет постоянное значение q_w во всех поперечных сечениях. Это приближение выполняется, когда слой Кнудсена покрывает все поперечное сечение.

Интегрирование выражения для общего теплового потока с учетом (3) и (4) позволяет получить, следующее выражение для аномальной теплопроводности в виде [13]

$$\mathrm{k_a} = \frac{k_0}{8Kn^2}(1 + 4CKn)\,, \qquad (5)$$

При высоких значениях Kn аномальная теплопроводность в соответствии полученным уравнением линейно убывает в зависимости Kn, что хорошо согласуется с экспериментальным данным [2]. Нетрудно заметить, что без включения граничного теплового потока (т. е. при C = 0), аномальная теплопроводность будет уменьшаться квадратично по отношению к обратной величине числа Кнудсена, что противоречит экспериментальным данным.

Теоретическая модель (5) для аномальной теплопроводности не выполняется всякий раз, когда характерный размер *R* по порядку величины совпадает с параметром l_{ph}, то есть, если $R \approx l_{ph}$. Тогда, для незначительных значений теплового потока на стенках объемный профиль теплового потока можно записать в следующем виде

$$\mathrm{q_b}(\mathrm{r}) = k(1 - \frac{Jo(ir/l_{ph})}{Jo(iR/l_{ph})})\,, \qquad (6)$$

где $Jo(z)$ - цилиндрическая функция Бесселя нулевого порядка.

Вклад пристеночного потока запишем в виде [14]

$$\mathrm{q_w}(\mathrm{r}) = -k(\frac{iJ_1(iR/l_{ph})}{Jo(iR/l_{ph})}Jo(ir/l_{ph}))\nabla T\,, \qquad (7)$$

где $J_1(z)$ -цилиндрическая функция Бесселя первого порядка. При преобразованиях использовали известное соотношение $J_0'(cz) = -cJ_1(cz)$.

Необходимо отметить, что значения функции $J_0(ir/l_{ph})$ и $iJ_1(iR/l_{ph})$ являются действительными. При этом минус в правой части уравнения (6) связан с необходимостью учета положительного значения потока q_w на границе. Следует отметить, что в этом же уравнении q_w зависит от радиального расстояния от стенки, поскольку теперь мы предполагаем, что радиус нано проволоки R сопоставим с длиной $\mathrm{l_{ph}}$.

Совместно интегрируя (6) и (7) вдоль поперечного сечения наноструктуры, получаем окончательное выражение для аномальной теплопроводности

$$k_a = k_0(1-2Kn(\frac{J_1(i/Kn)}{Jo(i/Kn)})^2(\frac{Jo(i/Kn)}{iJ_1(i/Kn)}+C))$$

(8)

Простые вычисления позволяют видеть, что всякий раз, когда *Kn* достигает высоких значений, (8) переходит в (5).

Рассмотрим распространение плоской температурной волны вида:

$$T(z,t) = \overline{T}_0(z) + \overline{T}_A e^{-i(\omega t - kz)}, \qquad (9)$$

где ω - волновая частота, κ - волновое число. $\overline{T}_0(z)$ - начальное значение, $\overline{T}_A$ -амплитуда волны.

Подставляя (9) в (2) получим дисперсионное уравнения для волнового числа:

$$k^2 = \frac{(\omega^2\tau_R - 1/\tau_r) - i\omega(1+\tau_R/\tau_r)}{\tau_R U_0^2},$$

(10)

Соответствующее выражение для фазовой скорости можно представить в виде

$$V_p = \frac{\omega}{\mathrm{X}}\sqrt{\frac{2}{1+\frac{\omega^2\tau_R\tau_r - 1}{\sqrt{(1+(\omega\tau_R)^2)(1+(\omega\tau_r)^2)}}}}$$

(11)

где $\mathrm{X} = \frac{((\omega^2 - \frac{1}{\tau_R\tau_r})^2 + \frac{\omega^2}{\tau_R}(1+\frac{\tau_R}{\tau_r})^2)^{\frac{1}{4}}}{U_0}$

На рисунке 9 приведены численно рассчитанные зависимости отношение τ_R / τ_{R_exc} от радиуса при постоянной температуре.

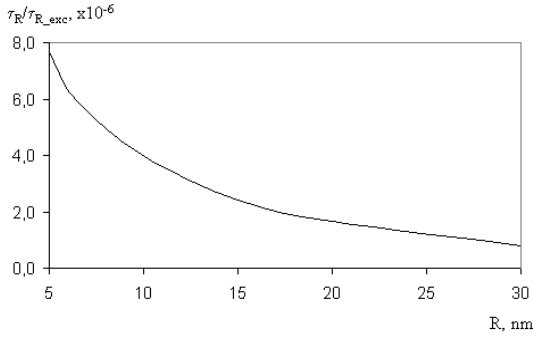


Рис. 9. Зависимость времен релаксации от температуры (T) верхний рисунок, где 1- R=5 нм, 2 – R=10 нм, 3 – R=20 нм, и от радиуса R при T=300 K

Из рисунка 9 видно, что с ростом R влияние граничных процессов постепенно уменьшается. При этом τ_R составляет ~10-6 τ_{R_exc} и, потому можно сделать вывод, что член $\frac{\dot{T}_t}{\tau_R}$ преобладает по отношению к $\frac{\dot{T}_t}{\tau_{R\_exc}}$.

4. Заключение

Численно получены решения гиперболического уравнения, описывающего аномальное распространение тепла в низкоразмерных системах для двух типов начального возмущения: узкого гиперболического и треугольного с выгнутой гипотенузой. Из результатов моделирования видно, в обоих случаях имеется ярко выраженный волновой фронт. Для параболической формы решение симметричное и в отрицательном и положительном направлении распространяются одинаковые волны. При этом затухание вблизи волнового фронта и нуля существенно отличаются. Во втором случае решение ассиметричное: волновой фронт в положительном направлении имеет гладкую вершину, отрицальном – ярко выраженный пик. Из-за ассиметричности волновых фронтов временная зависимость затухание отличается от случая параболического сигнала. Полученные результаты демонстрируют и волновую природу, и степенное затухание, что отличает их от решений классического уравнения теплопроводности. Свойства полученных решений могут быть использованы для анализа экспериментальных данных и выбора подходящей модели описания процесса.

Разработана математическая модель температурных волн применительно к кремниевым наноразмерным структурам с цилиндрической геометрией. В линейном режиме гидродинамика фононов основывается на гиперболического уравнении теплопроводности. При этом время релаксации из-за пристеночных столкновений учитывалось наряду с классическим временем релаксации из-за столкновений между фононами с использованием правила Маттиссена. Этот подход позволяет быстро тепловые свойства наноструктуры и может быть востребованным для того, чтобы выделить наиболее перспективные свойства наносистем.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № 0065-2019-0018 «Исследование и построение моделей и конструкций элементов микроэлектроники в расширенном диапазоне температур (-60oC…300oC)».

Anomalous Thermal Conductivity in Silicon Cylindrical Nanostructures

N.V. Masalsky

Abstract. . Solutions of the equation describing anomalous heat propagation in silicon fibers are discussed. Two cases of initial thermal perturbation are considered: a narrowly parabolic shape and a shape close to a sawtooth shape. It is shown that if the solution decays more slowly near the wavefront, pronounced maxima are observed, which can be detected experimentally. The obtained solutions demonstrate the wave nature and power-law attenuation, which distinguishes them from the solutions of the classical heat equation. A mathematical model of temperature waves is developed for silicon nanoscale structures with cylindrical geometry based on the hyperbolic heat equation. This approach allows us to study the thermal properties of nanostructures and may be in demand in order to identify the most promising properties of nanosystems.

Keywords: anomalous heat conduction, Knudsen number, MCV equation, heat wave

Литература

1. Y. Dong. Dynamical Analysis of Non-Fourier Heat Conduction and Its Application in Nanosystems. Springer, Berlin-Heidelberg-New York, 2016.

2. D.Y. Tzou. Macro- to Microscale Heat Transfer: The Lagging Behaviour. 2nd edn. Wiley, Chichester, 2014.

3. A. Sellitto, V. Cimmelli, D. Jou. Influence of electron and phonon temperature on the efficiency of thermoelectric conversion. “Int. J. Heat Mass Transf.”. V. 80 (2015), 344–352.

4. H. Ottinger. Beyond Equilibrium Thermodynamics. Wiley, New York, 2005.

5. Z. Zhang. Nano/Microscale Heat Transfer. McGraw-Hill, New York, 2007.

6. J.C. Maxwell. A Treatise on Electricity and Magnetism, 2nd edn. Oxford University Press, Cambridge, 1904.

7. C. Cattaneo. Sulla conduzione del calore. “Atti Sem. Mat. Fis. Univ. Modena”, V. 3 (1948), 83–101.

8. I. Ferain, C. A. Colinge, J. Colinge. Multigate transistors as the future of classical metal–oxide–semiconductor field-effect transistors. Nature. V. 479 (2011), 310–316.

9. B. Li, C. Chen, W. Yang, T. Li, C Pan, Y. Chen. Biomolecular recognition with sensitivity-enhanced nanowire transistor biosensor. “Biosensor and bioelectronics”, V. 45 (2013), 252-259.

10. H. Struchtrup. Macroscopic Transport Equations for Rarefied Gas Flows: Approximation Methods in Kinetic Theory - Interaction of Mechanics and Mathematics. Springer, New York, 2005.

11. A.M. Gheitaghy, M.R. Talaee. Solving hyperbolic heat conduction using electrical simulation. “J. Mechanical Sci. And Technology”, V.27 (2013), 3885-3891.

12. R.A. Guyer, J.A. Krumhansl. Thermal conductivity, second sound and phonon hydrodynamic phenomena in nonmetallic crystals. “Phys. Rev.”. V. 148 (1966), 778–788.

13. A.J.H. McGaughey, E.S. Landry, D.P. Sellan, C.H. Amon. Size-dependent model for thin-film and nanowire thermal conductivity. “Appl. Phys. Lett.”, V. 99 (2011), 131904-131908

14. A. Sellitto, F. Alvarez, D Jou. Geometrical dependence of thermal conductivity in elliptical and rectangular nanowires. “Int. J. Heat Mass Transfer”, V. 55 (2012), 3114–3120.

Методы подавления помех в цепях электропитания систем на кристалле

О.В. Сердин[1], С.Е. Серяков[2]

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

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

Аннотация. Статья содержит рекомендации по разработке систем питания СнК. В статье рассмотрены основные источники шума в цепях питания. Особое внимание уделено пульсациям, возникающим в результате скачков потребления СнК, общему импедансу цепи питания и топологии печатной платы. Разобраны основные рекомендации по созданию топологии печатной платы с минимальным импедансом цепи питания.

Ключевые слова: Печатная плата, импеданс цепи питания, конденсатор, помеха, Cadence Sigrity.

1. Введение

Организация системы питания современных систем на кристалле (СнК) является сложной задачей, возникающей на этапе разработки печатной платы (ПП). Вследствие постоянного увеличения количества транзисторов увеличивается и рассеиваемая мощность СнК. В такой ситуации обязательной является качественная система питания, позволяющая передать достаточное количество энергии кристаллу в короткий промежуток времени. Низкий импеданс цепи питания (Z_{pdn}) позволяет избежать просадок напряжения и удержать напряжение питания на допустимом уровне, при резких перепадах токов потребления СнК.

В структурном виде, полная цепь питания СнК представлена на рис.1 [1].

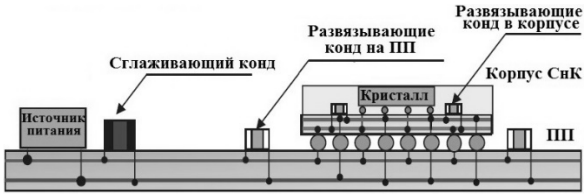


Рис. 1. Структура системы питания.

Упрощенная принципиальная схема цепи питания кристалла представлена на рис. 2.

Сглаживающие (Bulk) конденсаторы находятся около источника питания, и должны обладать большим объёмом и низким последовательным сопротивлением. Исходя из этого, их габариты значительно больше, чем габариты развязывающих (Decoupling) конденсаторов.

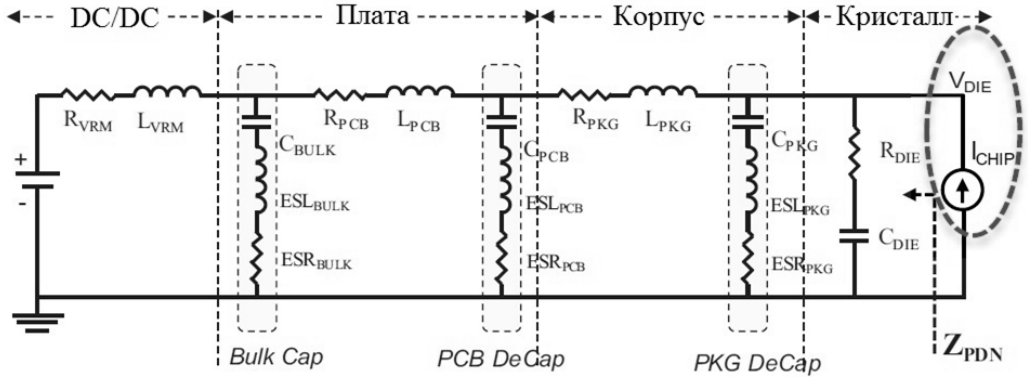


Рис. 2. Упрощенная принципиальная схема цепи питания кристалла

Из рисунков 1 и 2 видно, что система питания СнК складывается из четырех составляющих: источника питания, печатной платы, корпуса и кристалла. Решающее влияние на импеданс цепи питания оказывают конденсаторы, находящиеся на печатной плате и в корпусе микросхемы.

В данной работе, наибольшее внимание уделено снижению импеданса цепи питания при проектировании топологии ПП. Это обусловлено тем, что соблюдение некоторых рекомендаций по подбору структуры и сечения печатной платы, расположению конденсаторов и переход-

ных отверстий является первым шагом к качественной системе питания СнК. При этом соблюдение данных рекомендаций не требует дополнительных финансовых затрат на закупку конденсаторов или специализированных САПР.

2. Основные источники шума в цепи питания

На печатной плате и в корпусе присутствуют паразитные индуктивности, которые приводят к резкому увеличению импеданса на определенных частотах. Самым ярким примером является индуктивность корпуса микросхемы. Итоговое напряжение на кристалле изменяется за счет просадки напряжения из-за постоянного тока (DC), и высокочастотного шума, вызванного резкими скачками тока потребления кристалла (AC). С DC просадками помогает справить обратная связь источника питания, а AC шум зависит от импеданса цепи питания и характера потребления СнК.

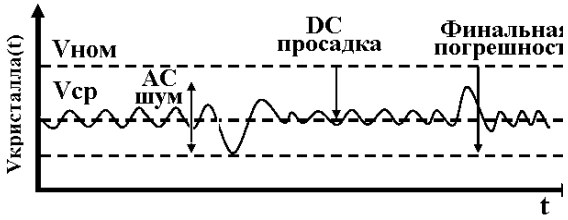


Рис 3. Напряжение питания на кристалле.

AC шум на кристалле состоит из трех составляющих: короткие пики напряжения, характеризующиеся значениями отклонения от номинального $V_{ном} – V_{пик}$, просадка напряжения – $V_{прос}$ и пульсации напряжения $V_{пульс}$, представленными на рис. 4.

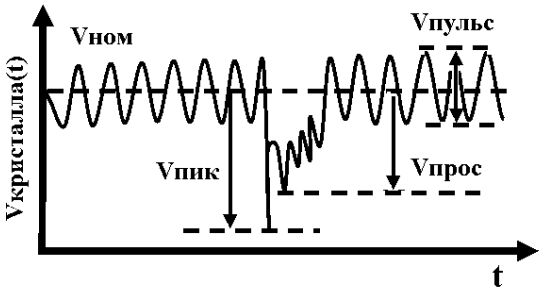


Рис 4. Типы скачков напряжения на кристалле.

Данные типы шумов и их основные характеристики представлены в табл. 1. Знание их временных характеристик позволяет эффективно бороться с каждым типом помех.

Таблица 1. Типы помех в цепи питания

Параметр	Пик	Просадка	Пульсация
Время эффекта	Наносекунды	Микросекунды	Микросекунды
Частота эффекта	Средние-высокие частоты	Низкие частоты	Зависит от частоты переключения DC/DC
Конденсаторы для борьбы с данным эффектом	Развязывающие конденсаторы на нагрузке	Крупные (bulk) конденсаторы на источнике, развязывающие конденсаторы на нагрузке	Крупные конденсаторы на источнике
Уровень помех	≈ 3%	≈ 3%	≈ 2%

Помехи, вызванные резкими скачками тока потребления, за счет их малой длительности, успешно купируются развязывающими конденсаторами. Второй тип – просадка напряжения – происходит на низких и средних частотах и снижается увеличением ёмкости bulk конденсаторов на выходе импульсного источника питания. Пульсации, вызванные переключением ключей источника питания, устраняются bulk конденсаторами с минимальным последовательным сопротивлением, подключенными с минимальной паразитной индуктивностью и максимально точным подбором обвязки преобразователя.

Пульсации, вызванные изменением характера потребления СнК, устраняются за счет раз-

вязывающих конденсаторов, находящихся на печатной плате и в корпусе микросхемы. Уровень пульсаций напряжения питания определяется согласно закону Ома, исходя из возможного мгновенного перепада тока потребления ($I_{maxtransient}$) и полного импеданса цепи питания. Если в исходных данных не оговорено иного, для большинства СнК в качестве мгновенного перепада тока потребления можно использовать половину от значения максимального тока потребления (I_{max}).

$$I_{maxtransient} = 1/2 * I_{max} \quad (1)$$

Зная допустимое колебание напряжения в данной цепи и теоретически максимальный мгновенный перепад тока потребления, можно вычислить целевой импеданс Z_{target}, являющийся в первом приближении максимально допустимым значением сопротивления цепи питания [2].

$$Z_{target} = V_{пик}/I_{maxtransient} \quad (2)$$

Разбору темы шумов в цепи питания, их источникам и методу их контроля с помощью целевого импеданса, посвящен ряд литературных источников.

Важным элементом метода подавления помех в цепях электропитания является их измерение. На примере V_{ripple} импульсного DC/DC, в работе [3] описана методика измерения уровня помех с помощью анализатора спектра и цепи стабилизации импеданса сети (LISN). Дополнительно показана методика оценки уровня помех при помощи осциллографа, не используя узкоспециализированное оборудование. Также показана необходимость оценки уровня помех, распространяемых изделием еще на этапе его разработки.

В работе [4] разобраны причины возникновения шума импульсного DC/DC, контуры токов, протекающих при переключении ключей в преобразователе, а также схемы фильтрации данных помех с конкретными значениями затуханий. Статья [5] содержит более подробный разбор топологии импульсного понижающего DC/DC. В статье [5] указаны наиболее важные с точки зрения шумоподавления цепи и компоненты обвязки импульсного DC/DC. Подробно описаны рекомендации по их расположению и подключению.

В статье [6] подробно описан метод целевого импеданса и его оптимизация на примере микросхемы DCP010505BP. Пошагово разобран процесс определения целевого импеданса микросхемы и подбора конденсаторов для трёх диапазонов частот. Данные диапазоны определяются предварительным моделированием импеданса микросхемы на всём диапазоне частот, до подбора развязывающих конденсаторов. Наиболее значительные пики на графике импеданса требуют соответствующего конденсатора для их устранения.

Примеры использования метода целевого импеданса и способы оптимизации количества развязывающих конденсаторов исследуются в работах [7], [1] и [8]. В статье [7] рассматривается метод оптимизации количества развязывающих конденсаторов с помощью оценки реального потребления ядра процессора и увеличения времени возрастания напряжения при переключении источника питания. В статье [1] для улучшения метода целевого импеданса используют расширенную модель потребления процессора и его тактового сигнал. Что позволяет более точно установить требования к импедансу цепи питания и избежать излишних вложений. В статье [8] для уточнения целевого импеданса микросхемы используют расширенную модель источника питания, анализируя напряжение на его выходе.

При построении графика импеданса цепи питания от частоты необходимо определиться, какова эффективная частота используемой микросхемы. В большинстве случаев эта частота находится в диапазоне от 50 до 100 МГц. При рассмотрении импеданса на частотах выше эффективной частоты, анализируется не превышение целевого импеданса микросхемы, а отсутствие его резких пиков, которые повлияют на высокочастотные сигналы на печатной плате. Корректные подбор и использование развязывающих конденсаторов с учетом проблемных участков импеданса цепи питания являются наиболее эффективным методом уменьшения импеданса.

Однако подбор конденсаторов, во-первых, требует от разработчика наличия САПР для моделирования импеданса, а во-вторых, влияет на стоимость конкретного проекта за счет возможности наличия избыточных конденсаторов. В данной ситуации разработчик сталкивается с необходимостью соблюдения баланса между улучшением характеристик изделия и финансовыми возможностями. В современные САПР также заложена возможность автоматического подбора развязывающих конденсаторов с учетом ограничений, выставленных по стоимости или по количеству номиналов. В данной статье больший акцент делается на топологию печатной платы, так как в большинстве случаев соблюдение рекомендаций по трассировке цепей питания и подбору сечения является первым шагом к качественной системе питания ИС, который не требует использования специализированной дорогостоящей САПР. Реализовав, на сколько это возможно, качественную топологию печатной платы с точки зрения импеданса цепей питания, можно переходить к подбору развязывающих

конденсаторов. В такой ситуации финальные количество и стоимость элементов будут значительно меньше, чем если изначально делать ставку на уменьшения импеданса цепи питания лишь путем установки подходящих конденсаторов.

3. Работа с топологией печатной платы

Разработка сечения печатной платы является одним из первых этапов. В современных модулях количество слоев, отведенных под полигоны цифровой землю, обычно является достаточно большим для обеспечения необходимого импеданса сигнальных линий и экранирования сигнальных слоев между собой. Количество слоев, отведенных под полигоны питания, значительно меньше. Парное расположение в соседних слоях полигона питания и полигона цифровой земли добавляет значительную емкость цепи питания, что значительно понижает импеданс данной цепи. Дублирование полигона питания в других слоях платы и создание дополнительных пар слоев питание-земля также в значительной степени уменьшает импеданс [3], однако требуются дополнительные слои, что зачастую невозможно. На первом этапе разработки сечения есть возможность изменения ширин диэлектриков между слоями меди или же изменения порядка слоев таким образом, чтобы добиться максимально эффективно работающей связки полигонов питания и земли. На рис. 5 представлены графики конечного импеданса для различных толщин диэлектрика на тестовой топологии подключения процессора в корпусе BGA898 с использованием небольшого набора 0,1 мкф и 4.7 мкф конденсаторов 0402 на bottom под процессором. Графики получены с использованием САПР Cadence Allegro Sigrity.

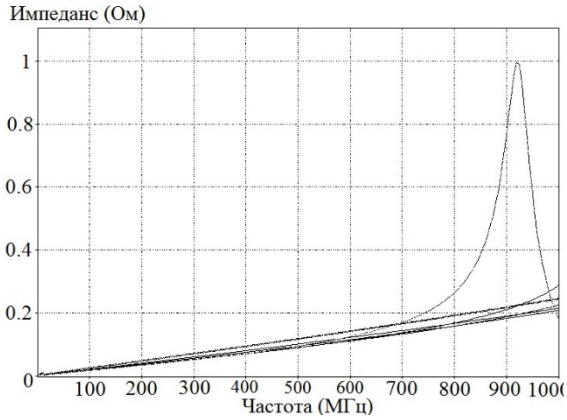


Рис 5. Зависимость импеданса цепи питания от толщины диэлектрика.

При уменьшении толщины диэлектрика между полигонами питания и земли уменьшается и уровень импеданса на всех частотах. Однако при переходе от 0.15 мм к 0.1 мм начинает смещаться пик импеданса вниз по частотной оси. Наиболее явно он виден на частоте 920 МГц при теоретической толщине диэлектрика 0.05 мм. Отсюда можно сделать вывод, что оптимальной является толщина диэлектрика от 0.15 мм до 0.3 мм. В этом случае значение импеданса на всех частотах относительно мало и можно не опасаться возникновения пика импеданса на частотах до 1-2 ГГц.

За счет внутренней структуры печатной платы возможно целенаправленное создание эффективных встроенных в печатную плату конденсаторов. Данный метод требует отдельного разбора и описан в статье [9]. Однако это часто невозможно на ПП с большим коэффициентом заполнения и плотной трассировкой.

Очевидным фактом является то, что чем ближе источник питания будет находиться к потребителю, тем меньше изначально будет импеданс цепи. Однако также интуитивно понятным является то, что необходимо стараться избегать нарушения целостности и монолитности полигона питания из-за наличия других сигналов на плате и необходимости использовать переходные отверстия. В противном случае импеданс увеличится. На самом деле наличие отверстий в полигоне сказывается только на DC характеристиках и на плотности тока. На рис. 6 показаны графики импеданса при целых полигонах питания и земли только с учетом переходных у пинов BGA процессора и при наличии в полигоне отверстий 0.8 мм в сетке с шагом 2,2 мм., 1,4 мм, 1 мм.

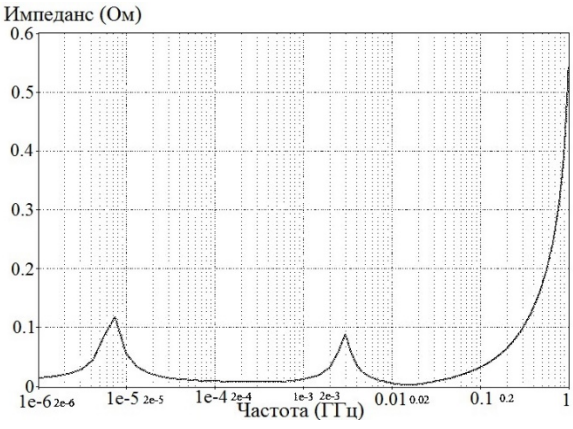


Рис 6. Зависимость импеданса цепи питания от целостности полигона.

Графики полностью совпадают, сливаясь в один. Следовательно, нет смысла пытаться уменьшать импеданс цепи питания оптимизируя количество отверстий в полигоне.

Очень важен способ подключения конденсатора к переходному отверстию. Чем шире и короче дорожка от конденсатора к переходному, тем меньше его индуктивность. Для наглядной демонстрации взята тестовая топология с небольшим набором конденсаторов 0.1 мкф и 4.7 мкф. В первом варианте эти 30 конденсаторов подключены с максимальной индуктивностью и, следовательно, импедансом. Образовавшийся путь тока показан пунктиром на рис. 7(а).

а) б)

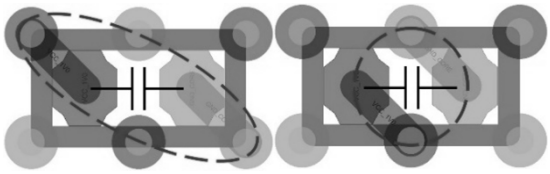


Рис 7. Неоптимальное подключение конденсаторов: а) диагональное подключение конденсатора к переходным отверстиям; б) подключение конденсатора только к ближайшим переходным отверстиям.

Во втором варианте, путь протекания токов в конденсаторе уменьшен (рис. 7(б)).

В следующем примере (рис. 8(а)), максимально использована возможность соединения конденсатора с соседними переходными отверстиями. В таком случае будет задействован максимально короткий путь тока из всех образовавшихся.

а) б)

Рис 8. Оптимальное подключение конденсаторов: а) задействованы все переходные отверстия; б) конденсатор установлен на переходные отверстия.

В последнем варианте (рис. 8(б)), индуктивность дорожек до конденсатора исключена, а конденсатор установлен непосредственно на переходные отверстия. Данный вариант доступен по согласованию с производством, так как возникает проблема утечки припоя в переходное отверстие.

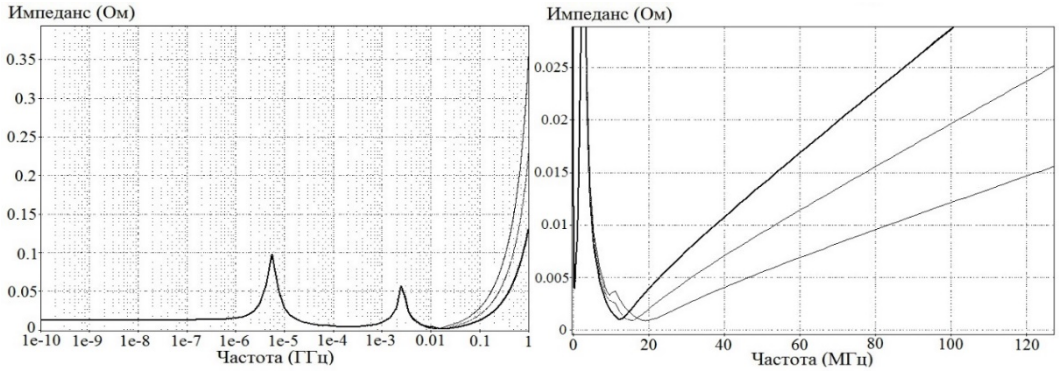


Рис 9. Зависимость импеданса цепи питания от топологии конденсатора: а) полный график до 1 ГГц, логарифмическая ось частот; б) импеданс цепи питания до 120 МГц, где находится эффективная частота большинства микросхем, линейная ось частот.

На представленных графиках (рис. 9) (общий график – логарифмическая частотная ось, увеличенный график – линейная частотная ось): 1 и 2 вариант подключения конденсаторов сливаются в одну линию, следовательно, на частотах до 1 ГГц не умеет значения, каким способом подключены конденсаторы, если финальная длина дорожек от конденсатора до переходного идентична. Однако такой результат возможен по причине того, что модель конденсатора в САПР подключается к центру пина. В реальности же конденсатор немного меньше посадочного места и подключение по 2 варианту имеет меньшую длину дорожек, а, следовательно, и индуктивность от реального конденсатора до переходного отверстия и является приоритетным.

Подключения с максимальным использованием доступных переходных отверстий или установкой конденсаторов непосредственно на переходные отверстия дает значительное уменьшение импеданса цепи. Эффективная частота большинства корпусов ИС находится в пределах 50 - 100 МГц. Именно на частотах от 15 МГц индуктивность дорожек подключения конденсаторов вносит наибольший вклад. Лучшей ситуацией является установка конденсаторов непосредственно на переходные отверстия. Однако если такой технологической возможности нет, рекомендуется сократить до минимума длину дорожек от переходного до площадки конденсатора. Если есть возможность удвоить или утроить количество соединений конденсатора с пере-

ходными отверстиями, то это дополнительно повысит эффективность конденсатора. При подключении двух и более конденсаторов к одному переходному отверстию критично важно располагать их в порядке увеличения номинала (уменьшения их рабочих частот).

Дальнейшее уменьшение кривой импеданса цепи питания рекомендуется проводить за счет добавления новых конденсаторов на печатную плату и корректировки номиналов старых. Известным фактом является то, что необходимо подбирать конденсаторы с наименьшим сопротивлением на самых проблемных частотах графика импеданса.

3.1. Развязывающие конденсаторы.

Имея график распределения импеданса цепи питания в интересующей полосе частот, разработчик имеет возможность подбирать конденсаторы с наименьшим комплексным сопротивлением в проблемных частотах. Конденсаторы с большим номиналом имеют меньшее сопротивление на низких частотах (рис. 10), однако при увеличении частоты, вследствие паразитной индуктивности, их сопротивление увеличивается. Исходя из этого, важно использовать несколько видов конденсаторов. Когда сопротивление одного конденсатора увеличивается при увеличении частоты, включается другой конденсатор, поддерживая общий импеданс цепи на низком уровне [10].

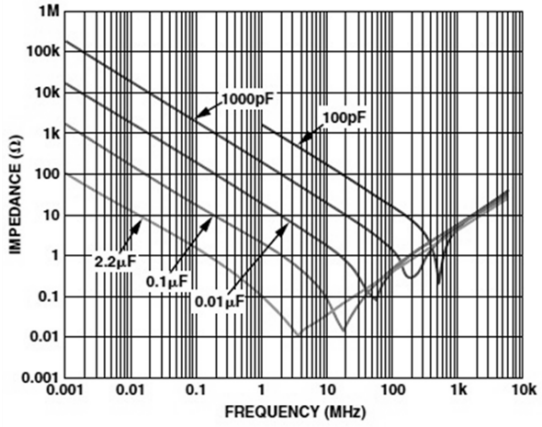


Рис. 10. Распределение комплексного сопротивления конденсаторов по частотам.

3. Заключение

В настоящее время, требования к цепям питания СнК ужесточаются, также усовершенствуются методы разработки качественной системы питания. В данной статье рассмотрены некоторые источники помех в цепи питания и их основные характеристики. Скачки напряжения, вызванные переключением ключей DC/DC преобразователя, а также просадки напряжения устраняются достаточной ёмкостью bulk конденсаторов на выходе преобразователя и индуктивностью, соответствующей заданному режиму. Необходимая ёмкость конденсаторов определяется исходя из тока потребления в данной цепи. Также важно обеспечить наименьшую петлю тока от выхода DC/DC через данные ёмкости.

Одной из важнейших проблем являются пульсации напряжения, вызванные резким перепадом тока потребления ИС. Для определения и контроля качества системы питания используется метод общего импеданса цепи питания. Требуемый уровень импеданса цепи определяется исходя из тока потребления и контролируется в заданной для конкретной микросхемы эффективной частоте. Используя набор конденсаторов различных номиналов, установленных как можно ближе к потребителю, возможно добиться максимального результата по уменьшению общего импеданса цепи питания.

Дополнительно для упрощения задачи уменьшения импеданса за счет установки конденсаторов возможно придерживаться следующих рекомендаций: размещать DC/DC преобразователь как можно ближе к потребителю; располагать слой с полигоном питания и слой с полигоном земли попарно, уменьшая толщину диэлектрика между ними; подключать развязывающие конденсаторы максимально короткими и широкими проводниками, по возможности устанавливая их непосредственно на переходные отверстия.

Работа выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № 0065-2019-0004.

Methods of Ripple Suppression for SoC Power Delivery Networks

O.V. Serdin, S.E. Seryakov

Abstract. The article contains recommendations for the development of SoC power systems. The article discusses the main sources of noise in power delivery network. Particular attention is paid to ripples resulting from surges

in consumption of SoC, the impedance of the power supply circuit, and the topology of the PCB. The main recommendations for creating a PCB topology with a minimum impedance of the power delivery network are discussed.

Keywords: PCB, impedance, capacitor, ripple, Cadence Sigrity

Литература

1. Jun Xu , Siqi Bai, Kartheek Nalla, Mike Sapozhnikov. Power Delivery Network Optimization Approach using an Innovative Hybrid Target Impedance, IEEE. 2019, p. 211-216.

2. Guang Chen and Dan Oh. Improving the Target Impedance Method for PCB Decoupling of Core Power. 2014 Electronic Components & Technology Conference.

3. Перевод: Евгений Карташов. Измерение напряжения помех импульсного DC/DC-стабилизатора. Сборник технических статей Wurth Elektronik, 2016-2018 гг., с. 119-121.

4. Roland van Roy. Application Note: Reducing EMI in buck converters. AN045 – January 2016 Richtek.

5. Application Note: PCB Layout Techniques of Buck Converter. 2012 ROHM Co., Ltd. No. 60AN066E Rev.003 oct. 2017.

6. Guangzhao Li, Li Zhaia,Huiyuan Fenga, Hongtao Gu. Optimization design method of decoupling capacitor in PCB hardware of electric vehicle controller. The 8th International Conference on Applied Energy – ICAE2016, Energy Procedia 105 (2017), p. 3201 – 3206.

7. Guang Chen and Dan Oh. Improving the Target Impedance Method for PCB Decoupling of Core Power. 2014 IEEE Electronic Components & Technology Conference, с. 566-571.

8. Guang Chen, Ahmed Abou-Alfotouh, Zhiwei Liu, Mostafa Shabban, Dan Oh. Optimization of PCB PDN design using enhanced VRM model. IEEE. 2014 Electronic Components & Technology Conference, с. 845-849.

9. Mihai Catalin Arva, Eugen Diaconescu. Modeling of stackable embedded capacitors into the PCB layers. Technical and Physical Problems of Electrical Engineering, 10-12 September 2015

10. John Ardizzoni. A practical guide to high-speed printed-circuit-board layout. Analog Dialogue technical journal 39-09, September 2005.

Математическая модель для анализа разработки нефтяных месторождений с учетом интерференции ячеек заводнения

И.В. Афанаскин[1], А.В. Королев[2], А.А. Глушаков[3]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, ivan@afanaskin.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, alexandre.korolev@mail.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, stormwww@yandex.ru

Аннотация. В работе подробно разбирается математическая модель для анализа разработки нефтяных месторождений с учетом интерференции ячеек заводнения. Эта модель основана на уравнениях сохранения объемов нефти и воды в стандартных условиях для ячейки заводнения. Перетоки между ячейками учитываются с помощью источникового слагаемого. Приводятся результаты тестирования предлагаемой модели, которая демонстрирует хорошую прогнозную точность. На фактическом примере показана необходимость учета интерференции ячеек.

Ключевые слова: ячейки заводнения, анализ разработки нефтяных месторождений, математическая модель

1. Введение

Для эффективной разработки нефтяных месторождений необходимо постоянно проводить контроль и регулирование разработки, анализ разработки. Одним из основных способов разработки нефтяных месторождений в России является заводнение. В таком случае для анализа разработки активно применяется метод ячеек заводнения [1-5]. Для этого пласт разбивается на блоки, содержащие нагнетательные и добывающие скважины. Как правило, нагнетательные скважины располагаются внутри блоков, а добывающие скважины могут располагаться как внутри, так и на границе. Границы блоков проводят по рядам добывающих скважин, границам залежи, непроницаемым границам (разломам) внутри залежи и пр. Скважины, расположенные на границе блоков, принадлежат нескольким блокам сразу. Поэтому их добычу необходимо разделять между соседними ячейками заводнения. Для оценки влияния нагнетательных скважин на добывающие их обычно располагают внутри блоков вдали от их границ.

В классической постановке в качестве математической модели для метода ячеек заводнения используется комбинация уравнения материального баланса и метода характеристик заводнения [2-5]. Также могут использоваться более физически обоснованные математические модели [1]. Однако в этих случаях не учитывается интерференция ячеек заводнения, т.е. переток нефти и воды из одной ячейки в другую. Это может приводить к значительным ошибкам, поскольку в неоднородных по простиранию пластах между соседними ячейками заводнения могут создаваться существенные перепады пластового давления, способствующие значительным перетокам нефти и воды между ними. Эти перепады давления могут обуславливаться не только разницей в фильтрационно-емкостных свойствах, но и индивидуальными особенностями реализуемых систем разработки внутри соседних ячеек заводнения (расстановка скважин, соотношение между нагнетательными и добывающими скважинами, расположение относительно водонефтяного контакта, режимы эксплуатации скважин и др.).

Многие мелкие и некоторые средние месторождения разрабатываются при естественном водонапорном режиме (т.е. не благодаря энергии закачиваемой воды, а благодаря энергии воды в законтурной водоносной области). Для анализа их разработки применяются подходы, похожие на метод ячеек заводнения. Нефтяные пласты таких месторождений также разбиваются на блоки, показатели разработки которых анализируется отдельно друг от друга. При этом учет перетоков между блоками является особенно важным, поскольку источник воды не распределен по залежи по какой-то системе, как в случае с заводнением, а находится за ее пределами и вода продвигается от границы залежи к ее центру.

Настоящая работа посвящена подробному разбору математической модели для анализа разработки нефтяных месторождений с учетом интерференции ячеек заводнения и ее тестированию.

2. Математическая модель

Сформулируем математическую модель, описывающую процесс разработки нефтяного месторождения в рамках концепции ячеек заводнения. Запишем уравнения сохранения объемов нефти и воды в стандартных условиях для ячейки заводнения:

$$F_i h_i \frac{d}{dt}\left(\frac{m_i S_{o,i}}{B_{o,i}}\right) = -\sum_{l\in N_i} q_{o,l}(t) - \sum_{j\in K_i} q_{o,ij}(t), \quad (1)$$

$$F_i h_i \frac{d}{dt}\left(\frac{m_i S_{w,i}}{B_{w,i}}\right) = -\sum_{l\in N_i} q_{w,l}(t) - \sum_{j\in K_i} q_{w,ij}(t) + \sum_{l\in M_i} q_{iw,l}(t) + q_{a,i}(t), \quad (2)$$

где F_i, h_i, m_i, $S_{o,i}$ и $S_{w,i}$, $B_{o,i}$ и $B_{w,i}$, N_i, M_i и K_i – площадь, эффективная толщина, пористость, нефтенасыщенность и водонасыщенность, объемные коэффициенты нефти и воды, множество добывающих скважин, множество нагнетательных скважин и множество соседних ячеек i-ой ячейки заводнения соответственно, t – время, $q_{o,l}(t)$ и $q_{w,l}(t)$ - дебиты нефти и воды l-ой скважины соответственно, $q_{o,ij}(t)$ и $q_{w,ij}(t)$ - перетоки нефти и воды соответственно из i-ой ячейки заводнения в j-ую, $q_{iw,l}(t)$ - расход закачиваемой воды l-ой скважины, $q_{a,i}(t)$ - приток воды из законтурной водоносной области в i-ую ячейку заводнения.

Запишем балансовое соотношение для насыщенностей нефти и воды:

$$S_{o,i} + S_{w,i} = 1. \quad (3)$$

Пусть жидкости слабосжимаемые, а пласт упругий, тогда:

$$m_i = m_{0,i}\left[1 + C_{r,i}\left(P_i - P_0\right)\right], \quad (4)$$

$$B_{\alpha,i} = B_{\alpha,0}\left[1 - C_\alpha\left(P_i - P_0\right)\right],\ \alpha = o, w, \quad (5)$$

где $m_{0,i}$ - пористость i-ой ячейки заводнения при опорном давлении P_0, $C_{r,i}$ - сжимаемость породы пласта i-ой ячейки заводнения, P_i - пластовое давление в i-ой ячейки заводнения, $B_{\alpha,0}$ - объемный коэффициент фазы $\alpha=o,w$ при опорном давлении P_0, C_α - сжимаемость фазы $\alpha=o,w$.

С учетом соотношений (3)-(5) уравнения (1) и (2) могут быть преобразованы с применением подхода, аналогичного изложенному в [6], к виду:

$$F_i h_i m_{0,i}\left[C_r + (C_w - C_o)S_i + C_o\right]\frac{dP_i}{dt} = -B_{o,0}\left[\sum_{l\in N_i} q_{o,l}(t) + \sum_{j\in K_i} q_{o,ij}(t)\right] - B_{w,0}\left[\sum_{l\in N_i} q_{w,l}(t) + \sum_{j\in K_i} q_{w,ij}(t) - \sum_{l\in M_i} q_{iw,l}(t) - q_{a,i}(t)\right], \quad (6)$$

$$F_i h_i m_{0,i}\left[\frac{dS_i}{dt} + (C_r + C_w)S_i\frac{dP_i}{dt}\right] = -B_{w,0}\left[\sum_{l\in N_i} q_{w,l}(t) + \sum_{j\in K_i} q_{w,ij}(t) - \sum_{l\in M_i} q_{iw,l}(t) - q_{a,i}(t)\right], \quad (7)$$

где введено обозначение для насыщенности $S_i \equiv S_{w,i}$.

Уравнение (6) - уравнение для давления, а уравнение (7) - уравнение для насыщенности.

В качестве граничных условий принимаются условия непротекания. Влияние скважин и законтурной водоносной области учтено через источниковые слагаемые.

В качестве начальных условий задаются значения насыщенности и пластового давления в начальный момент времени: $P_i(t=0)=P_{i,0}$ и $S_i(t=0)=S_{i,0}$.

Уравнения (6) и (7) решаются методом конечных разностей по схеме Эйлера с учетом начальных и граничных условий. На каждом шаге по времени сначала определяется давление, а затем насыщенность. При этом конечно-разностные соотношения для давления и насыщенности имеют вид:

$$P_i^{n+1} = P_i^n - \Delta t^{n+1} \times \left\{ \frac{B_{o,0}\left[\sum_{l\in N_i} q_{o,l}^n + \sum_{j\in K_i} q_{o,ij}^n\right]}{F_i h_i m_{0,i}\left[C_r + (C_w - C_o)S_i^n + C_o\right]} + \frac{B_{w,0}\left[\sum_{l\in N_i} q_{w,l}^n + \sum_{j\in K_i} q_{w,ij}^n - \sum_{l\in M_i} q_{iw,l}^n - q_{a,i}^n\right]}{F_i h_i m_{0,i}\left[C_r + (C_w - C_o)S_i^n + C_o\right]} \right\},$$

$$S_i^{n+1} = S_i^n - \Delta t^{n+1} \frac{B_{w,0}}{F_i h_i m_{0,i}} \times$$

$$\times \left[\sum_{l \in N_i} q_{w,l}^n + \sum_{j \in K_i} q_{w,ij}^n - \sum_{l \in M_i} q_{iw,l}^n - q_{a,i}^n \right] +$$

$$+ (C_r + C_w) S_i^n \left(P_i^{n+1} - P_i^n \right),$$

где n - номер шага по времени, Δt^{n+1} - шаг по времени, который рассчитывается так, чтобы изменения давления и насыщенности за один шаг по времени не превышали заданных критических пороговых значений [7].

Дебиты скважины по фазам определяются следующим образом:

$$q_{\alpha,l}^n = \lambda_{\alpha,l}^n PI_l^n \left(P_i^n - P_{well,l}^n \right), \ \alpha = o, w,$$

$$\lambda_{\alpha,l}^n = \frac{k_{r\alpha,l}^n}{\mu_\alpha B_{\alpha,i}^n}, \ \alpha = o, w,$$

где $\lambda_{\alpha,l}^n$ - коэффициент подвижности флюида $\alpha=o,w$ для l-ой скважины, PI_l^n - индекс продуктивности l-ой скважины (независящая от флюида часть коэффициента продуктивности), $P_{well,l}^n$ - забойное давление l-ой скважины, $k_{r\alpha,l}^n$ - относительная фазовая проницаемость (ОФП) флюида $\alpha=o,w$ для l-ой скважины, μ_α - динамическая вязкость флюида $\alpha=o,w$.

Забойное давление определяется, как:

$$P_{well,l}^n = P_l^n - q_{liq,l}^n / \left[\left(\lambda_{o,l}^n + \lambda_{w,l}^n \right) PI_l^n \right],$$

где $q_{liq,l}^n$ - дебит l-ой скважины по жидкости.

Как правило для управления скважинам задается забойное давление либо дебит жидкости (сумма дебитов нефти и воды). Если задан дебит жидкости, необходимо сначала рассчитать забойное давление, а потом дебиты нефти и воды. В случае расположения скважины на границе двух и более ячеек заводнения суммарный дебит этой скважины (по воде, нефти и жидкости) вычисляется как сумма притоков к скважине из всех дренируемых ячеек.

Расход закачиваемой воды следующим образом связан с забойным давлением:

$$q_{iw,l}^n = \lambda_{w,l}^n \left(S_i^n = 1 - S_{owcr} \right) PI_l^n \left(P_{well,l}^n - P_l^n \right),$$

где S_{owcr} - насыщенность пласта остаточной нефтью.

Переток фазы $\alpha=o,w$ из i-ой ячейки заводнения в j-ую определяется, как:

$$q_{\alpha,ij}^n = \lambda_{\alpha,ij}^n PI_{ij} \left(P_i^n - P_j^n \right), \ \alpha = o, w,$$

$$\lambda_{\alpha,ij}^n = \frac{k_{r\alpha,ij}^n}{\mu_\alpha B_{\alpha,ij}^n}, \ \alpha = o, w,$$

где $\lambda_{\alpha,ij}^n$ - коэффициент подвижности флюида $\alpha=o,w$ при перетоке из i-ой ячейки заводнения в j-ую, PI_{ij} - коэффициент перетока (аналог индекса продуктивности), $k_{r\alpha,ij}^n$ - ОФП флюида $\alpha=o,w$ для перетока (выбирается из $k_{r\alpha,i}^n$ и $k_{r\alpha,j}^n$ по принципу «вверх по потоку»), $B_{\alpha,ij}^n$ - объемный коэффициент флюида $\alpha=o,w$ для вычисления перетока, определяется как среднее арифметическое между $B_{\alpha,i}^n$ и $B_{\alpha,j}^n$. При расчете перетоков между ячейками целесообразно задавать максимальное значение ОФП по нефти и нулевое значение ОФП по воде при насыщенности менее насыщенности на фронте вытеснения. Таким образом удается избежать мгновенного перетока воды между ячейками сразу после начала закачки, так как такой эффект представляется физически неверным.

Приток воды из законтурной водоносной области в i-ую ячейку заводнения определяется, как:

$$q_{a,i}^n = \alpha_i \lambda_{a,i}^n PI_a \left(P_a^n - P_i^n \right), \ \sum_{i \in N_a} \alpha_i = 1,$$

$$\lambda_{a,i}^n = \frac{1}{\mu_w B_{w,a}^n},$$

где $\lambda_{a,i}^n$ - коэффициент подвижности воды в законтурной области, α_i - доля площади контакта i-ой ячейки заводнения с законтурной водоносной областью по отношению ко всей площади контакта нефтяной залежи и водоносной области, PI_a - индекс продуктивности законтурной области, P_a^n - давление в законтурной области, N_a - множество ячеек заводнения, контактирующих с законтурной областью.

Давление в законтурной области определяется с помощью уравнения сохранения объема воды в стандартных условиях в законтурной области способом, аналогичным приведенному выше для пластового давления в ячейке заводнения:

$$P_a^{n+1} = P_a^n - \Delta t^{n+1} \frac{B_{w,0} \sum_{i \in N_a} q_{a,i}^n}{F_a h_a m_{0,a} \left(C_{r,a} + C_w \right)}, \quad (8)$$

где F_a, h_a, $m_{0,a}$ и $C_{r,a}$ - площадь, эффективная толщина, пористость при опорном давлении P_0 и сжимаемость породы законтурной водоносной области соответственно.

Начальное условие для решения уравнения (8): $P_a(t=0)=P_{a,0}$.

При адаптации приведенной модели по фактическим показателям работы скважин в первую очередь модификации подлежат: параметры зависимости ОФП от насыщенности, комплекс $F_i h_i m_{0,i}$ или C_r, комплекс $F_a h_a m_{0,a}$ или C_a, PI_l^n, PI_{ij}, PI_a. Функции ОФП для разных скважин, а также для перетоков между разными наборами сосед-

них ячеек как правило различаются между собой, порой - весьма существенно.

3. Тестирование предложенной модели

Для тестирования предложенной математической модели для анализа разработки нефтяных месторождений с учетом интерференции ячеек заводнения была создана фильтрационная модель в коммерческом гидродинамическом симуляторе Rubis компании Kappa Engineering [8].

Модель имеет размеры 600x1200 м и разбита на 2 ячейки заводнения размерами 600x600 м каждая, рис. 1. В разработке участвуют 6 добывающих скважин P1, P2, P3, P4, P5, P6 и 2 нагнетательных скважины I1 и I2. Законтурная водоносная зона отсутствует.

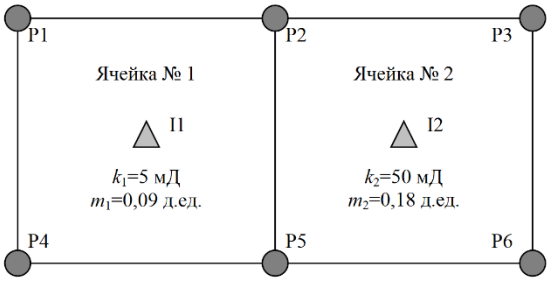


Рис. 1. Схема расстановки скважин и границ в модели

В первую ячейку заводнения входят скважины P1, P2, P4, P5 и I1, а во вторую - P2, P3, P5, P6 и I2. Таким образом скважины P2 и P5 являются общими для этих ячеек, следовательно их добычу необходимо разделять между ячейками. Чтобы показатели разработки по симметрично расположенным внутри ячеек добывающим скважинам не совпадали нагнетательные скважины были сдвинуты относительно центров ячеек на 35 м на северо-запад, т.е. в сторону верхнего левого угла на рис. 1.

Эффективная толщина пласта 10 м. Абсолютная проницаемость первой ячейки 5 мД, пористость 0,09 д.ед. Абсолютная проницаемость второй ячейки 50 мД, пористость 0,18 д.ед. Сжимаемость породы пласта $4{,}5 \cdot 10^{-5}$ 1/бар.

Начальное пластовое давление 350 бар. Начальная водонасыщенность 0,2 д.ед.

Объемный коэффициент воды 1,01 $м^3/м^3$, динамическая вязкость воды 0,31 мПа·с, сжимаемость воды $4{,}9 \cdot 10^{-5}$ 1/бар.

Объемный коэффициент нефти 1,15 $м^3/м^3$, динамическая вязкость нефти 0,36 мПа·с, сжимаемость нефти $1{,}05 \cdot 10^{-4}$ 1/бар, давление насыщения нефти газом 124 бар.

Исходные функции ОФП для нефти и воды приведены на рис. 2. Насыщенность связанной водой 0,2 д.ед., ОФП по нефти в этой точке 1,0 д.ед. Насыщенность остаточной нефтью 0,2 д.ед., ОФП по воде в этой точке 0,5 д.ед. Показатель степени для ОФП нефти и воды 2. Насыщенность на фронте вытеснения 0,67 д.ед. При адаптации модели ячеек заводнения эти ОФП модифицировались.

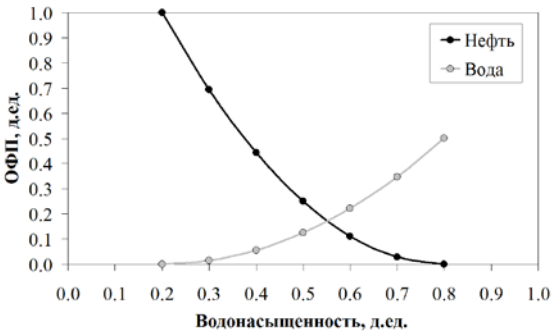


Рис. 2. Исходные функции ОФП

Результаты расчетов с помощью Rubis приняты в качестве «фактических», т.е. опорных данных. При этом первые 15 лет приняты в качестве истории разработки, затем следуют 15 лет прогнозного периода.

Нагнетательные скважины работают при постоянном забойном давлении 400 бар.

Добывающие скважины в течение истории работают при постоянном расходе жидкости: P1 - 10 $м^3$/сут, P2 - 40 $м^3$/сут, P3 - 30 $м^3$/сут, P4 - 10 $м^3$/сут, P5 - 40 $м^3$/сут, P6 - 30 $м^3$/сут. Первый прогнозный вариант предполагает эксплуатацию скважин в установившемся режиме. Второй прогнозный вариант предполагает увеличение дебитов жидкости всех добывающих скважин в два раза.

На рис. 3-5 приведены изменение пластового давления и насыщенности, а также перетоки нефти и воды из ячейки № 2 в ячейку № 1 согласно модели ячеек заводнения. Видно, что перетоки сопоставимы с дебитами жидкости скважин ячейки № 1, следовательно их влияние на разработку значительно и они должны обязательно учитываться при моделировании.

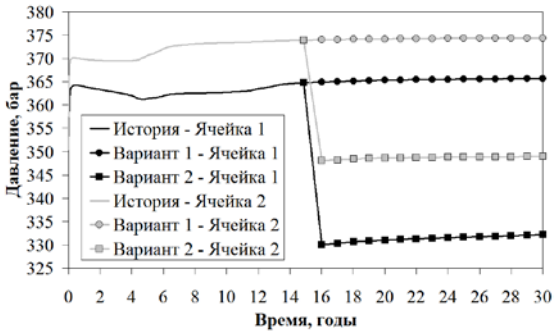


Рис. 3. Изменение пластового давления согласно модели ячеек заводнения

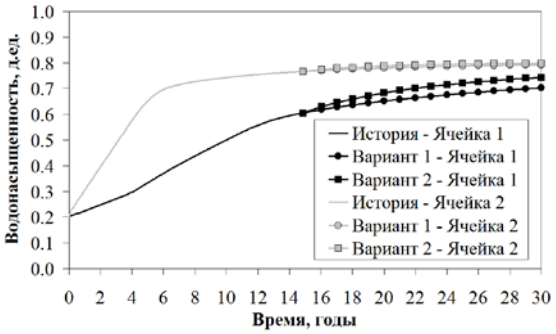


Рис. 4. Изменение насыщенности согласно модели ячеек заводнения

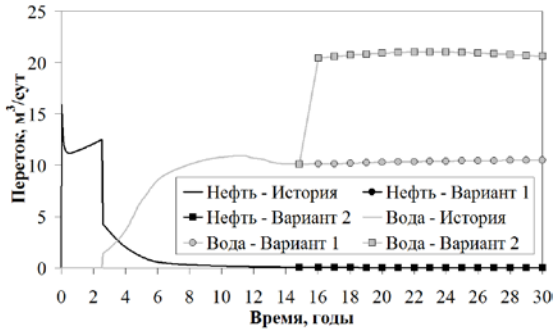


Рис. 5 Перетоки нефти и воды из ячейки № 2 в ячейку № 1 согласно модели ячеек заводнения

На рис. 6-11 приведены дебит нефти, обводненность и накопленная добыча нефти по скважинам за исторический период, а также для первого и второго прогнозного вариантов. Видна хорошая адаптация модели ячеек заводнения на результаты расчетов в Rubis. На конец истории разработки расхождение в накопленной добыче нефти по каждой скважине составляет менее 0,5 %. Максимальная ошибка в накопленной только за прогнозный период добыче нефти составляет 2,6% для первого варианта разработки и 17,1% для второго. Минимальная ошибка в накопленной только за прогнозный период добыче нефти составляет 0,6% для первого варианта разработки и 7,0% для второго. Наиболее низкая точность прогноза соответствует скважинам Р1 и Р4, так как они целиком находятся в низкопроницаемой ячейке и на конец истории разработки их обводненность продолжает активно расти. Наиболее высокая точность прогноза соответствует скважинам Р3 и Р6, так как они целиком находятся в высокопроницаемой ячейке и на конец истории разработки их обводненность имеет высокое значение, а скорость ее роста замедлилась. Средняя точность прогноза соответствует скважинам Р2 и Р5, которые обводняются из двух источников (двух нагнетательных скважин), так как находятся на границе между двумя ячейками заводнения.

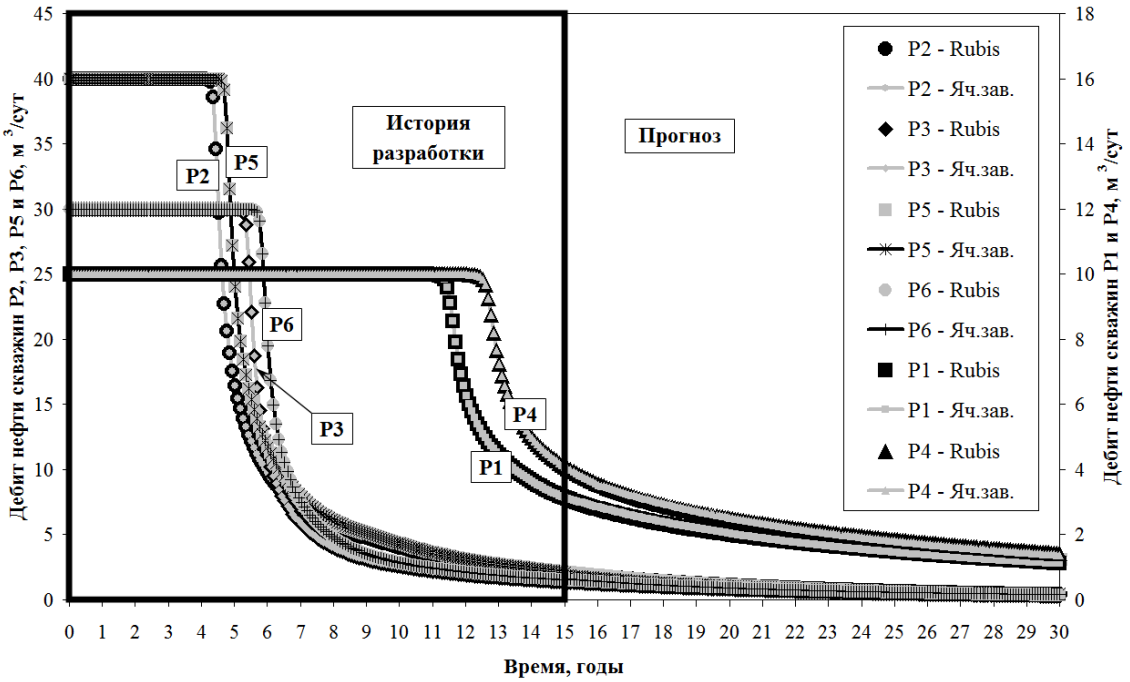


Рис. 6. Дебит нефти по скважинам за исторический период и для первого прогнозного варианта

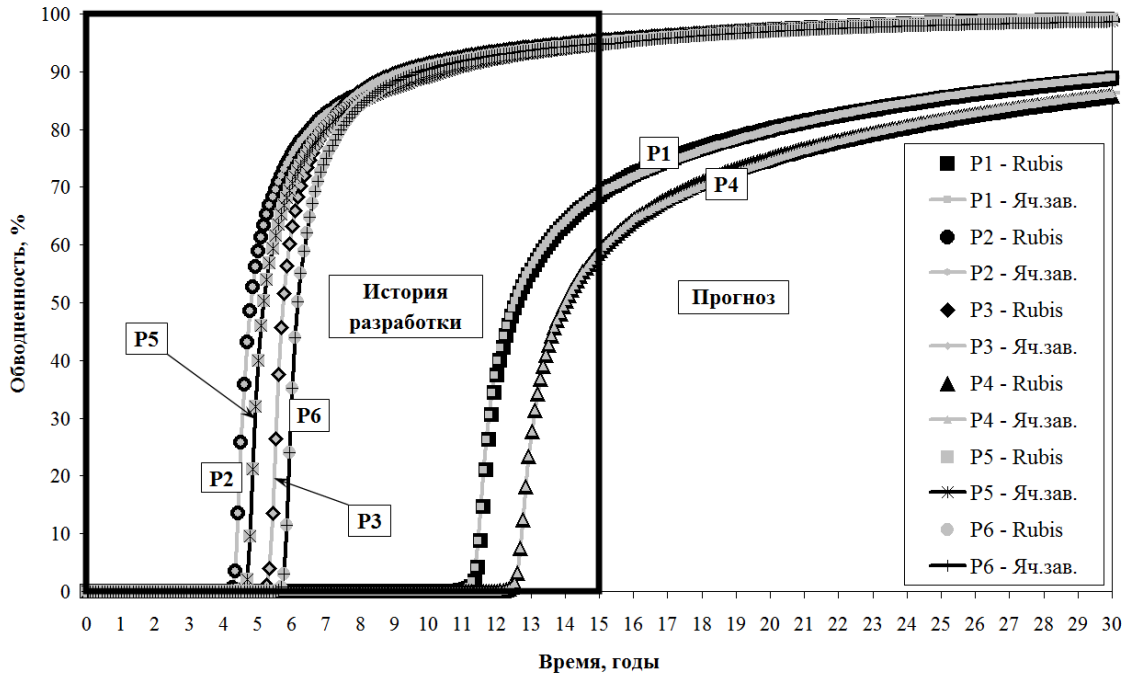


Рис. 7. Обводненность по скважинам за исторический период и для первого прогнозного варианта

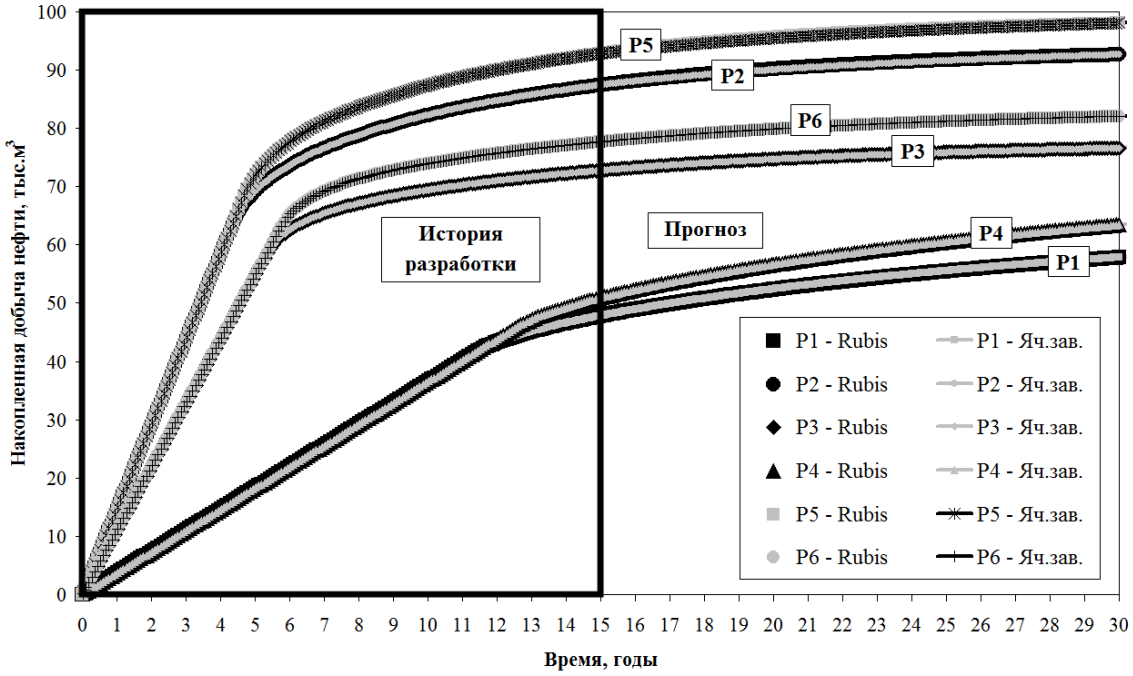


Рис. 8. Накопленная добыча нефти по скважинам за исторический период и для первого прогнозного варианта

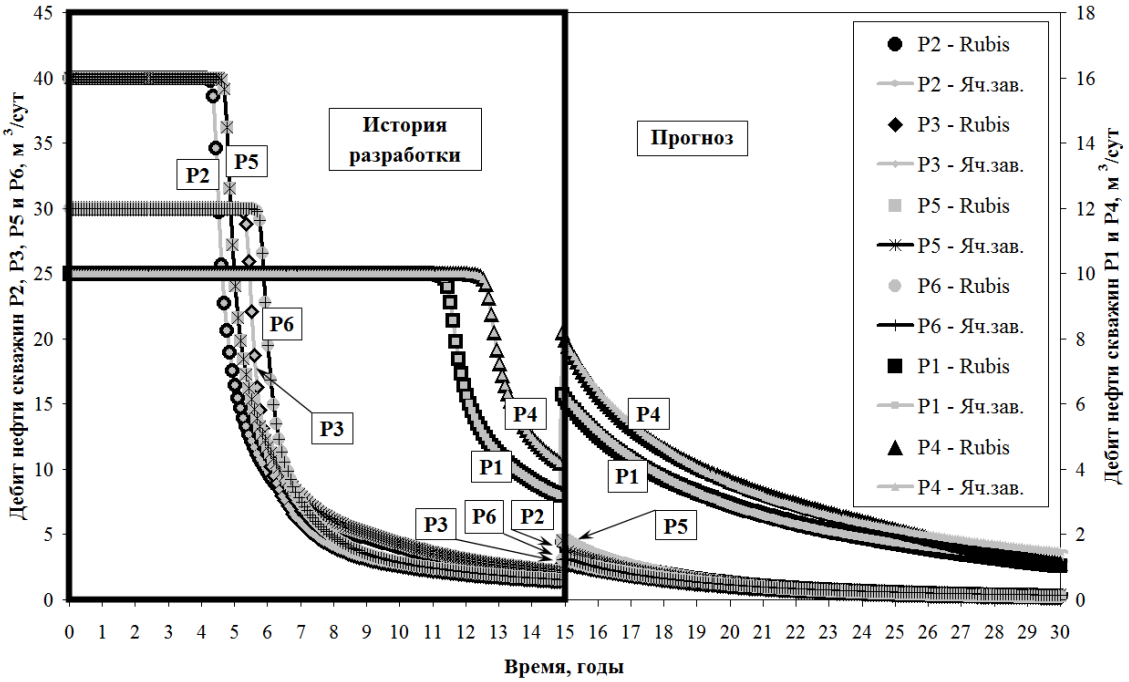


Рис. 9. Дебит нефти по скважинам за исторический период и для второго прогнозного варианта

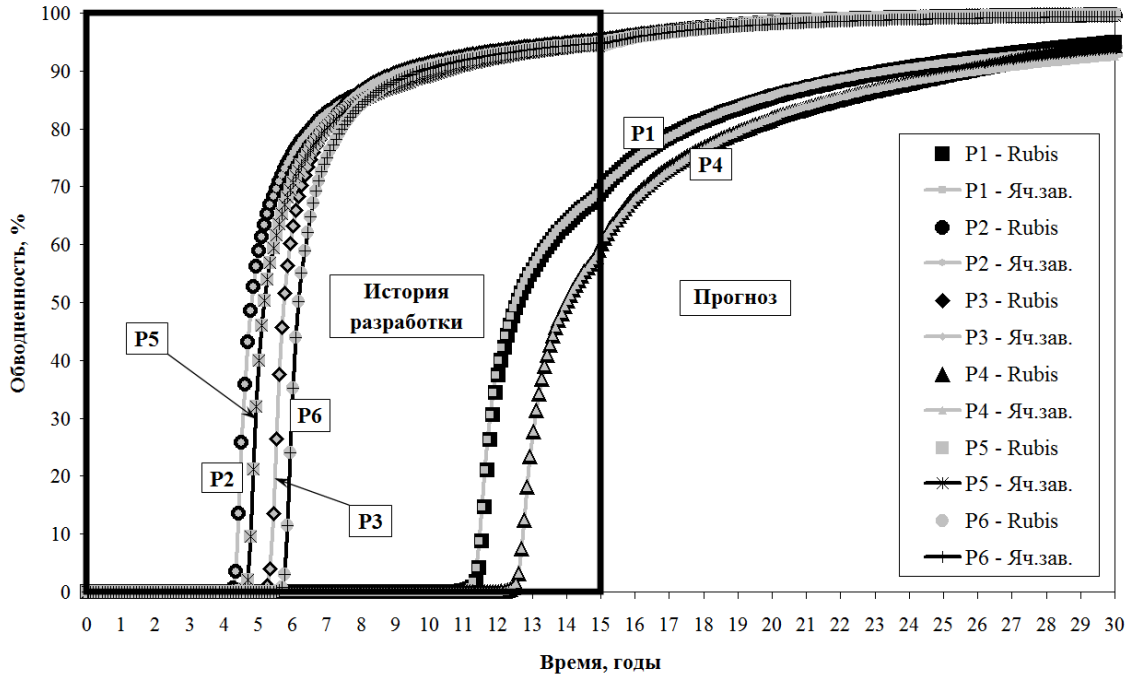


Рис. 10. Обводненность по скважинам за исторический период и для второго прогнозного варианта

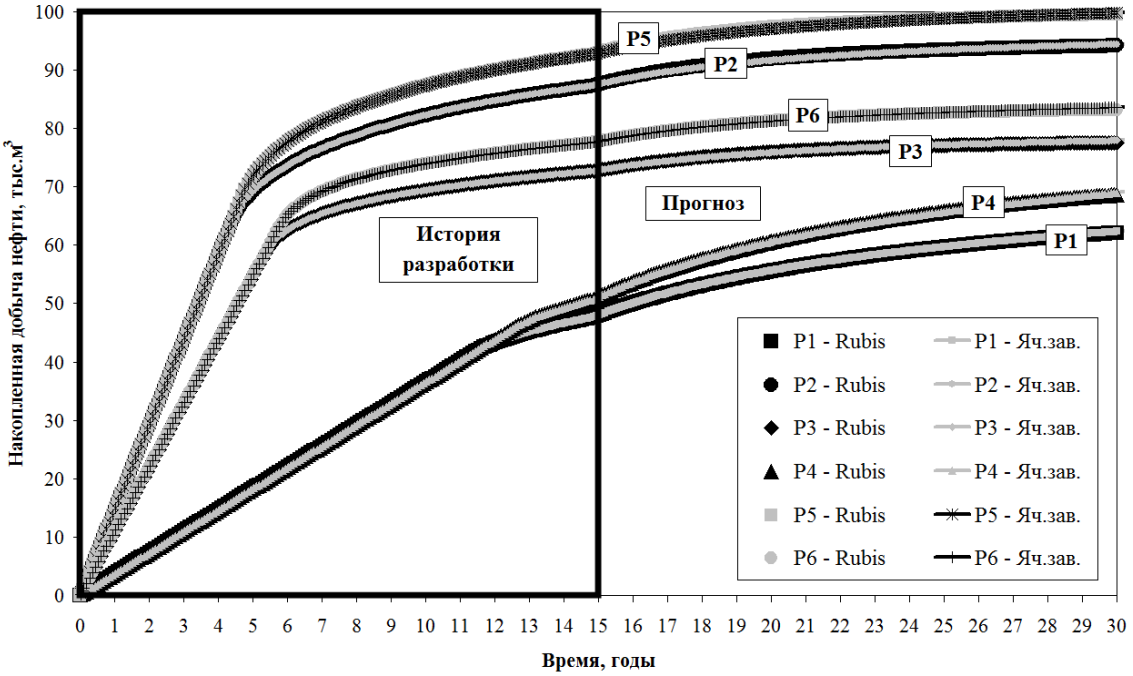


Рис. 11. Накопленная добыча нефти по скважинам за исторический период и для второго прогнозного варианта

3. Заключение

В работе предложена математическая модель для анализа разработки нефтяных месторождений с учетом интерференции ячеек заводнения. Модель учитывает работу добывающих и нагнетательных скважин, приток воды из законтурной области, перетоки нефти и воды между ячейками заводнения. Модель основана на уравнениях сохранения объемов нефти и воды в стандартных условиях для ячейки заводнения и уравнении сохранения объема воды в стандартных условиях в законтурной области. Приток воды из законтурной области и перетоки нефти и воды между ячейками заводнения учитываются с помощью источниковых слагаемых. Таким образом данная модель является физически содержательной, что выгодно отличает ее от обычно используемой при анализе разработки методом ячеек заводнения связки уравнения материального баланса и метода характеристик вытеснения.

Приведены результаты тестирования предложенной модели. В качестве базы сравнения использована специально созданная в гидродинамическом симуляторе Rubis Kappa Engineering фильтрационная модель. Модель ячеек заводнения адаптировалась на условно принятый «исторический» период разработки, затем были рассмотрены два прогнозных варианта - разработка в сложившихся условиях и форсированный отбор. Получена удовлетворительная точность прогноза накопленной добычи нефти.

Приведены рассчитанные с помощью модели ячеек заводнения перетоки нефти и воды между ячейками. Показано, что перетоки сопоставимы с дебитами жидкости скважин ячейки с более низкими фильтрационно-емкостными свойствами, следовательно их влияние на разработку значительно и они должны обязательно учитываться при моделировании.

Работа выполнена при поддержке гранта РФФИ 18-07-00677 А.

Mathematical Model for Analysis of Oil Fields Development Taking into Account Waterflooding Cells Interfering

Ivan Afanaskin, Aleksandr Korolev, Aleksej Glushakov

Abstract. This paper presents detailed mathematical model for analysis of oil fields development taking into account waterflooding cells interference. The model is based on conservation equation for oil and water in waterflooding cells in standard conditions. The fluid flows between cells are presented in conservation equation as additional

source/sink terms. The testing results for described model are presented with good forecast accuracy achieved. The importance of cells interference terms is shown for practical case.

Keywords: waterflooding cells, oil field development analysis, mathematical model

Литература

1. И.В. Афанаскин. Адресная оценка эффективности реализуемых систем разработки нефтяных месторождений. «Геология, геофизика и разработка нефтяных и газовых месторождений», (2016), № 8, 44-54.

2. Управление разработкой пласта. «TGL Reservoir Engineering Group», 2010.

3. И.Ф. Хатмуллин, Е.И. Хатмуллина, А.Т. Хамитов и др. Технология решения типовых задач по оптимизации заводнения. «Техническая конференция SPE: Оптимизация заводнения на зрелых месторождениях. Тюмень, 26-28 ноября 2013 (труды)».

4. М. Наугольников, Н. Тепляков, М. Большаков. Cost-БФА: инструмент стоимостной оптимизации заводнения. «Российская нефтегазовая техническая конференция SPE. Москва, 15-17 октября 2018». SPE-191580-18RPTC-RU.

5. Д. Уолкотт. Разработка и управление месторождениями при заводнении. М., Юкос, 2001.

6. Р.М. Кац, Е.Р. Волгин, И.В. Афанаскин. Численное моделирование двухфазной фильтрации нефти и воды. «Труды НИИСИ РАН», Т. 4. (2014), № 2, 141-148.

7. Х. Азиз, Э. Сеттари. Математическое моделирование пластовых систем. М.-Ижевск, ИКИ, 2004.

8. O. Houze, D. Viturat, O.S. Fjaere and all. Dynamic Data Analysis. V 5.30.01. Kappa Engineering, 2020.

Практические результаты определения типов пласта коллектора и оценки проницаемости

А.А. Колеватов[1], Ю.Б. Чен-лен-сон[2], А.М. Гиацинтов[3], А.А. Глушаков[4]

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

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

[3] ФГУ ФНЦ НИИСИ РАН, Москва, Россия, algts@inbox.ru

[4] ФГУ ФНЦ НИИСИ РАН, Москва, Россия, stormwww@yandex.ru

Аннотация. В статье представлены практические результаты применения методики оценки проницаемости, а также, приводятся результаты анализа структуры пустотного пространства, необходимые при выполнении проектных работ. Выводы об интерпретационной модели двойной проницаемости, как о преобладающем механизме фильтрации флюидов в пласте, подтверждаются, в том числе, результатами гидродинамических исследований скважин.

Ключевые слова: оценка проницаемости, карбонатный трещиноватый нефтенасыщенный коллектор, гидродинамические исследования

1. Введение

На основе полученных ранее результатов, [1, 2], были выполнены практические работы с целью подтверждения теоретических предположений о свойствах пласта-коллектора, а также уточнения строения изучаемого объекта в части площадного распределения фильтрационно-емкостных свойств (ФЕС). Была проведена классификация типов пласта-коллектора в скважинах, не охваченных гидродинамическими исследованиями (ГДИ) с последующей оценкой проницаемости.

2. Предпосылки разработки методики оценки проницаемости

Основные предпосылки разработки методики оценки проницаемости карбонатных трещиноватых нефтенасыщенных коллекторов заключаются в следующем:

- несоответствие значений проницаемости, рассчитываемых по петрофизическим зависимостям керн-ГИС значениям проницаемости, определяемым по результатам ГДИ [1];
- наличие преимущественного механизма фильтрации флюидов к скважинам «двойная проницаемость», связанное со значительной трещинной составляющей [3] в определяемых величинах скин-фактора и проницаемости;
- результаты ГДИ, подтверждающие преимущественные перетоки через системы трещин разного порядка на большей части площади исследовавшегося нефтенасыщенного пласта-коллектора.

2.1. Подтверждение механизма фильтрации флюидов «двойная проницаемость»

Пласт-коллектор, информация по исследованиям которого анализировалась в рамках всей работы, условно можно представить в виде двух слоев, расположенных один над другим. В общем виде такой пласт коллектор может быть представлен по аналогии с рис. 1.

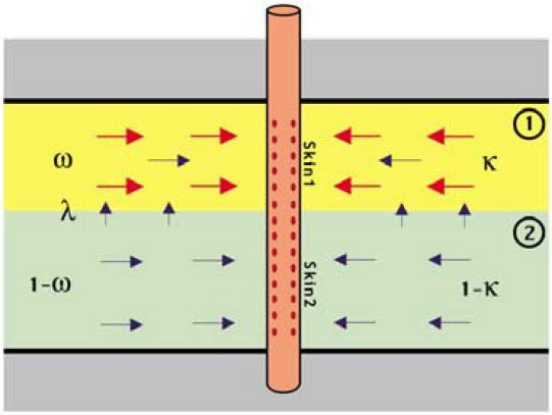


Рис. 1. Иллюстрация модели фильтрации «двойная проницаемость» [4,6]

Данный рисунок имеет следующий физический смысл – гидродинамическая связь двух нефтенасыщенных объектов через систему субвертикальных трещин разного масштаба, а также, через систему взаимосвязанных пор и ка-

верн. Наличие такой системы флюидопроводящих каналов означает, что даже при вскрытии скважиной только одного пласта со скважиной будут связаны оба пласта за счет перетоков.

В этом случае диагностический график [7, 8] в билогарифмических координатах будет иметь вид, как на рис. 2. Аналогичный вид диагностического графика наблюдался по результатам ГДИ в 10 скважинах исследовавшегося объекта.

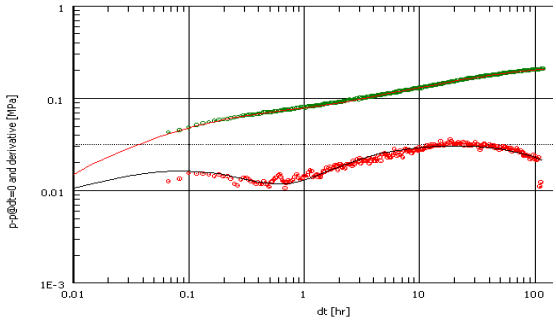


Рис. 2. Диагностический график полученный методом наилучшего совмещения для модели фильтрации флюида «двойная проницаемость»

2.2. Подтверждение трещинной составляющей, как одного из факторов определяющих механизм фильтрации «двойная проницаемость»

В рамках выполнения настоящего исследования изучались практические наработки специалистов по интерпретации результатов гидродинамических исследований. Рассчитанные по результатам ГДИ величины проницаемости и скин-факторов по каждой скважине, локализованные в зоне от -3.5 и менее, свидетельствуют о преимущественной связи скважин с нефтенасыщенным пластом-коллектором через систему трещин (рис. 3).

Рис. 3. Соотношение скин-фактора и проницаемости по результатам ГДИ

3. Практические результаты применения методики оценки проницаемости

В процессе анализа результатов промысловых геофизических исследований по некоторым вскрытым интервалам продуктивность установить не удалось. Однако, применение интерпретационной модели «двойная проницаемость» для интерпретации ГДИ показало, что в зоне дренирования некоторых таких скважин работали оба объекта. На рис. 4 представлено сравнение информации о продуктивности вскрытых пластов до и после интерпретации ГДИ.

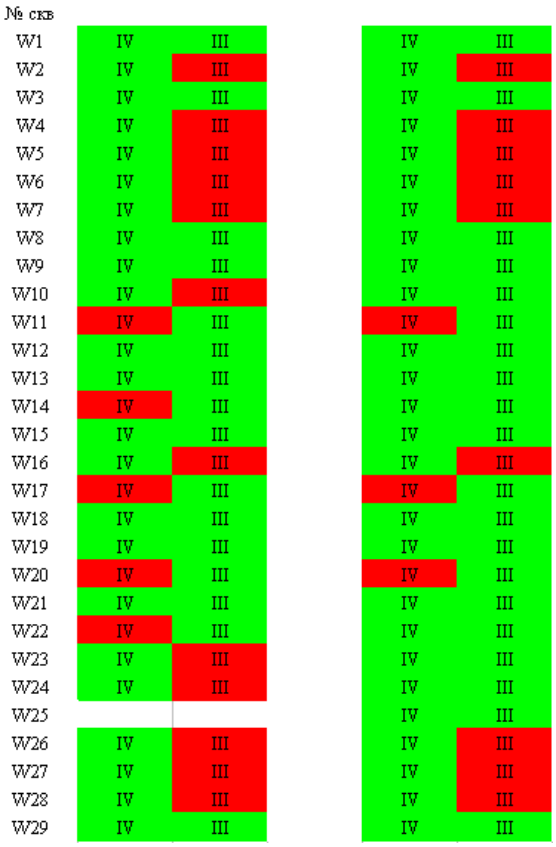


Рис. 3. Информация о продуктивности интервалов, вскрытых скважинами, до (слева) и после (справа) интерпретации ГДИ. Зеленым цветом обозначены интервалы с подтвержденной продуктивностью. Красным – интервалы с неподтвержденной продуктивностью.

В таблице 1 приведены результаты оценки проницаемости вскрытых скважинами интервалов в соответствии с методикой, представленной в статье [5]. Представленные результаты оценки проницаемости кроме числового параметра имели цель указать на различия в свойствах пластов коллекторов, которые могут иметь место в разных зонах. Отнесение пласта коллектора к

одному из типов может восполнить пробелы в информации, связанные с несовершенством существующих на сегодняшний день технологий изучения нефтегазоносных объектов.

Таблица 1. Классификация отдельных интервалов вскрытых скважинами и оценка проницаемостей

Скв. №	№ пачки	№ гр (по ГИС)	$\varphi_{ср.взв}$ д.ед	k,10^{-3} мкм^2 прогноз
30	IV	2	0.10	2400
31	IV	1	0.12	300-1950
32	III	2	0.09	500
33	IV	2	0.09	500
34	III	1	0.12	150-1000
35	IV	3	0.07	300
36	III	1	0.10	100-400
37	IV	3	0.07	350
38	III	3	0.08	1000
39	IV	3	0.08	1600
40	IV	3	0.07	600
41	IV	1	0.10	100-400
42	III	2	0.1	1500
43	IV	3	0.07	300
44	III	2	0.09	500

3. Заключение

Анализ статистической и качественной информации комплексных ГИС был использован в качестве основы для разработки методики идентификации структуры пустотного пространства выявленных групп нефтенасыщенных коллекторов.

Методика идентификации типов пласта коллектора и прогнозирования проницаемости создана на основе комплексирования статистической и качественной информации (результаты комплексных ГИС скважин).

С учетом обнаруженных граничных параметров породы пласта коллектора в скважинах, не исследовавшихся с помощью ГДИ, были отнесены к одному из 4-х типов, и сделана оценка проницаемости.

Методика идентификации типа пустотного пространства и оценки проницаемости может быть адаптирована для применения на объектах нефтегазодобычи, локализованых в карбонатных трещиноватых коллекторах. Полученные уравнения для прогноза проницаемости с учетом выявленного по ГИС типа пласта-коллектора не ограничиваются результатами конкретного исследования. Их количество может измениться при выявлении новых диагностических признаков. Это связано с несовершенством существующих технологий исследования месторождений в части получения однозначной информации о структуре пустотного пространства на удалении от скважины до нескольких метров и более.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта №18-07-00673 А, 2020 год.

Practical Results of Reservoir Type Identification and Permeability Estimation

A.A. Kolevatov, Yu.B.Chen-len-son, A.M. Giatsintov, A.A. Glushakov

Abstract. Article describes practical results of permeability estimation methodology application, as well as analysis results of porous media structure which is necessary for field development projects justification. Conclusions of dual permeability fluid flow regime, which prevails in particular reservoir, were confirmed also by well test results.

Keywords: permeability estimation, carbonate fractured oil saturated reservoir, well test

Литература

1. А.А. Колеватов, Ю.М. Штейнберг, Ю.Б. Чен-лен-сон, А.Г. Дяченко. Классификационные признаки типов нефтенасыщенных коллекторов по результатам гидродинамических исследований. «Труды НИИСИ», Т. 9 (2019), № 5, 59–62.

2. А.А. Колеватов, А.М. Гиацинтов, А.А. Глушаков, А.А. Егоров. Особенности карбонатных трещиноватых нефтенасыщенных коллекторов по данным геофизических исследований скважин Т. 9 (2019), № 5, 54–58.

3. A.C. Gringarten. Interpretation of Tests in Fissured and MultiLayered Reservoirs with Double-Porositi Behavior. Theory and Practice/Pet.Tech. – 1984. – April. – P.549-564.

4. О.Узе, Д.Витура, О. Фьярэ. Анализ динамических потоков, КАРРА Engineering, 2009.

5. А.А. Колеватов, Ю.Б. Чен-лен-сон, Ю.М. Штейнберг, А.Г. Дяченко. Методика оценки проницаемости нефтенасыщенных карбонатных трещиноватых коллекторов. Нефтепромысловое дело, № 11, 2020 г.

6. Т.А. Деева, М.Р. Камартдинов, Т.Е. Кулагина и др. Гидродинамические исследования скважин: анализ и интерпретация данных. Томск: ЦППС НД ТПУ, 2009. – 243 с.

7. Bourdet D. et al. A New Set of Type Curves Simplifies Well Test Analysis // World Oil. 1983. May. P. 95–106.

8. D. Bourdet Well Test Analysis: The Use of Advanced Interpretation Models. 2002. - 436 p.

Оценка технологической эффективности термогазового воздействия для баженовской свиты при закачке водовоздушной смеси

Д.Т. Миронов[1], А.А. Глушаков[2], П.В. Ялов[3]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, mdt1958@yandex.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, glushakoffal@yandex.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, petryalov@gmail.com

Аннотация. Рассматриваются вопросы математического моделирования трехфазной фильтрации в условиях применения термогазового метода увеличения нефтеотдачи (ТГ МУН). Описаны процессы при закачке водо-воздушной смеси для реализации влажного внутрипластового горения. Показаны характерные особенности данной технологии при закачке водо-воздушной смеси. Проведены расчеты и дана сравнительная оценку вариантов при закачке смеси с различными водовоздушным отношением и вариантов с закачкой обогащенной смеси при содержании кислорода в воздухе 30% и 50%.

Ключевые слова: Термогазовое воздействие, МУН, баженовская свита, закачка кислородсодержащей смеси (воздуха, водовоздушной смеси), кероген - органическое вещество, дренируемые и недренируемые породы, влажное внутрипластовое горение.

1. Введение

Работы по тепловому воздействию на керогенсодержащую породу баженовской свиты (БС) на участках Средне-Назымского месторождения в настоящее время проводятся в рамках испытания термогазового метода увеличения нефтеотдачи пластов (ТГМУН) [1].

В статье приводятся результаты оценки технологической эффективности термогазового воздействия при закачке водовоздушных смесей с различными соотношениями вытесняющих агентов. Основные подходы, реализованные при построении 2Д линейной модели с учетом подключения недренируемых интервалов (матрицы) в отложениях баженовской свиты, а также параметры процессов при ТГВ рассмотрены в предыдущих работах [2, 3]. В настоящей статье также проведена оценка применения модификаций ТГВ, таких как закачка воздуха с повышенным содержанием кислорода, что способствует повышению эффективности термогазового воздействия для недренируемых частей разреза баженовской свиты.

Говоря об основных подходах, реализованных в термогазовом воздействии надо отметить, что данный метод соединяет применение тепловых и газовых методов воздействия и впервые был предложен в 1971 году специалистами и учеными ВНИИнефть им. А.П. Крылова и имеет отечественный приоритет [4].

Окислительные процессы при ТГВ позволяют сформировать нефтевытесняющий агент, который содержит дымовые газы, углеводородные газы и широкие фракции легких углеводородов (ШФЛУ), выделяемые из пластовой нефти под влиянием теплового воздействия. Исследования показали, что объём закачки водо-воздушной смеси может быть ограничен созданием оторочки, составляющей 20-40% объема пор, а затем может быть осуществлен переход на другой вытесняющий агент [5].

При разработке отложений БС экономически целесообразно осуществлять совместный отбор легких нефтей из дренируемых зон, ресурсов УВ из недренируемых зон и формирования условий для генерации дополнительной синтетической нефти. Эффективная выработка запасов и ресурсов, включая добычу синтетических углеводородов, возможна при реализации методов увеличения нефтеотдачи, таких как закачка термогазовых смесей с определенными водовоздушным отношением и ряда других [6].

С целью увеличения охвата воздействием кероген-содержащих отложений БС необходимо расширение интервалов протекания процесса по разрезу и площади, что возможно при закачке термогазовых смесей с различным водовоздушным отношением (ВВО).

Важно отметить, что при увеличении содержания воды в смеси температура на фронте горения будет падать, что приведет к снижению эффективности. При высоких ВВО более 0,005 $м^3/нм^3$, что соответствует сверхвлажному горению, закачиваемая вода с определенного мо-

мента уже в жидком виде внедряется в зону горения, снижая температуру. В условиях керогенсодержащего коллектора это может привести к снижению температуры ниже температуры пиролиза и полное прекращение термодеструкции керогена. Поэтому оценка оптимальных параметров водовоздушных смесей очень важна при реализации термогазового воздействия на пласт.

Для моделирования процесса ТГВ на гидродинамическом симуляторе STARS (CMG Канада) была подготовлена 2D модель, включающая возможность вовлечение углеводородного сырья (УВС) из изначально недренируемых зон (матрицы) баженовской свиты в процессе термогазового воздействия. Подробно данная модель описана в работе [2].

Принципиальная схема, используемая в линейной модели «нагнетательная – добывающая скважины» представлена чередованием недренируемых кероген-содержащих непроницаемых пропластков и проницаемых. Недренируемые пропластки содержат основную часть органического вещества нефтематеринской породы – керогена, а приток флюидов из скважины и закачка агентов в пласт проводится из дренируемых интервалов, имеющих средние фильтрационно-емкостными и геохимические параметры, характерные для данных отложений

В модели применена композиционная модель пластовой нефти, которая содержит шесть углеводородных компонентов, от метановой до фракции C_{18+}, азот, углекислый газ, вода и кислород. Также добавлены три компонента - кероген, его твердый остаток (кокс) и синтетическая нефть, возникающие при термодеструкции (пиролизе) керогена. Учтены химические реакции окисления нефти и керогена, пиролиза керогена с образованием из него синтетической нефти и твердого остатка (кокса) и также горения твердого остатка.

Гидродинамические расчеты проводились в несколько этапов. Вначале проведены сравнительные расчеты с учетом и без учета эффектов гистерезиса ОФП. Сравнения вариантов показало, что при наличии чередующихся закачек воды и воздуха необходим учет изменений относительной фазовой проницаемости при дренаже и пропитке и учёт объёмов защемленного газа, как несмачивающейся фазы в системе газ-жидкость, и соответствующие результаты приведены в статье [7]. Неучет механизма влияет на оценку эффективности ТГВ, игнорирование этого может приводить к некорректным результатам.

Все дальнейшие расчеты на линейной модели проводились с учетом эффекта гистерезиса относительных фазовых проницаемостей.

Результаты расчетных вариантов, приведенные в настоящей статье, можно разделить на три этапа.

Первый этап включал сравнительные расчеты по закачке различных видов агентов и их комбинаций, включая: закачку инертного газа – азота (вариант 1), при котором тепловое воздействие на пласт не реализуется; вариант с постоянной закачкой сухого воздуха (вариант 2, базовый); вариант с закачкой водовоздушной смеси (Вариант 3).

Второй этап включал расчеты и сравнительную оценку вариантов при закачке смеси с различными ВВО (варианты №№ 3, 4, 5, 6) Закачка агентов с различными соотношениями воздуха и воды в смеси позволила определить наиболее оптимальное значение ВВО и также оценить необходимый объём оторочки воздуха и воды в зависимости от порового объёма пласта (PV) участка воздействия закачкой.

Третий этап включал расчеты и оценку вариантов (варианты №№ 4, 7, 8, 9) при закачке смеси различными циклами и периодами закачки агентов с оптимальным водовоздушным отношением, определенным по результатам расчета второго этапа.

Во всех вариантах соответствующим этапу 3 проводится циклическая закачка сухого воздуха и водовоздушной смеси с ВВО 0,0015 продолжительностью два года, (до создания оторочки с объемом агентов около 30-35 % порового объема участка воздействия), затем переход на закачку сухого воздуха. Первые два года идет чередование закачки водовоздушной смеси и сухого воздуха с различными периодами закачки воздуха (один - два месяца) и количеством циклов закачки.

Также проведен расчет вариантов с закачкой обогащенной смеси (содержание кислорода в воздухе повышено до 30% (вариант 10) и до 50% (вариант 11).

Кроме оценки общей эффективности процесса ТГВ, проведена количественная оценка таких важных составляющих накопленной добычи, как дополнительная добыча синтетической нефти (в стандартных условиях) из дренируемой и недренируемых зон и добыча УВ сырья (нативной нефти) из недренируемых (до проведения ТГВ) закрытых пустот в недренируемых интервалах (зонах) пласта.

При проведении расчетов в начальный период 2-х лет осуществлялась закачки водовоздушной смеси для формирования вытесняющей оторочки и затем проходил переход к закачке сухого воздуха для продвижения сформировавшейся оторочки. Во всех вариантах для корректной сравнительной оценки показателей разра-

ботки, режимы работы добывающих и нагнетательных скважин не менялись. Отбор нефти по скважине ведется при забойном давлении составляющим 0,9 от давления насыщения (16,6 МПа).

Исходные параметры вариантов при проведении ТГВ, закачке газовых агентов и смеси с различными ВВО и с чередованием циклов закачки приведены в таблице 1.

Таблица 1. Рассмотренные варианты закачки воздуха и водовоздушной смеси

№ варианта	ВВО, д.ед.(н/у)	Газосодержание смеси (ГС), %	Объем закачки смеси (в PV уч-ка ОПР,%	Среднее забойное давление (нагн/доб), МПа	Тип закачиваемого агента	Период и цикл закачки ТГВ, мес / (кол-во циклов)
1	2	3	4	5	6	7
1	0	100	0,403	20.0/15.0	N2	Постоянная закачка
2	0	100	0,283	30.0/15.0	Воздух	Постоянная закачка
3	0.001	75	0,303	40.0/15.0	Водо-воздушная смесь (ВВС)	Циклическая зак.(Нач.ВВО6мес) 2мес.СВ+1мес.ВВО
4	0.0015	67	0,319	40.0/15.0	ВВС	Циклическая зак.(Нач.ВВО6мес) 2мес.СВ+1мес.ВВО
5	0.002	60	0,326	43.0/15.0	ВВС	Циклическая зак.(Нач.ВВО6мес) 2мес.СВ+1мес.ВВО
6	0.005	38	0,325	45.0/15.0	ВВС	Циклическая зак.(Нач.ВВО6мес) 2мес.СВ+1мес.ВВО
7	0.0015	67	0,314	40.0/15.0	ВВС	Циклическая зак.(Нач.ВВО 3мес) 2мес.СВ+1мес.ВВО
8	0.0015	67	0,448	45.0/15.0	ВВС	Циклическая зак.(Нач.ВВО 6мес) 1мес.СВ+1мес.ВВО
9	0.0015	67	0,465	45.0/15.0	ВВС	Циклическая зак.(Нач.ВВО 6мес) 1мес.СВ+2мес.ВВО
10	0.0015	67	0,263	40.0/15.0	ВВС (30%O2)	Циклическая зак.(Нач.ВВО6мес) 2мес.СВ+1мес.ВВО
11	0.0015	67	0,201	40.0/15.0	ВВС (50% O2)	Циклическая зак.(Нач.ВВО6мес) 2мес.СВ+1мес.ВВО

2. Технологические показатели расчетных вариантов

2.1. Сравнительная оценка технологических показателей вариантов при закачке инертного газа (азота), сухого воздуха и водовоздушной смеси

На линейной модели, включающей матрицу и дренируемую зону, проведены расчеты вариантов с закачкой инертного газа - азота (вариант 1), закачка сухого воздуха (СВ) при ТГВ (вар.2) и закачка воздуха с ВВО 0,001 (вар.3).

Во всех расчетах закачка смеси с ВВО проводилась в течении первых двух лет для создания оторочки, с дальнейшим переходом на закачку сухого воздуха. Поэтому темпы отбора накопленной нефти и УВ после достижения и отбора вала нефти перед оторочкой и самой оторочки через добывающую скважину по вариантам с ВВО выравнивались. И дополнительный отбор осуществлялся, в основном, благодаря добыче из недренируемых зон.

Расчеты показали, что при закачке азота, как вытесняющего агента получено минимальное значение коэффициента извлечения УВ (рисунок 1), без подключения к разработке недренируемых зон, что определяется отсутствием окислительных реакций, характерных для закачки воздуха, при повышении температуры.

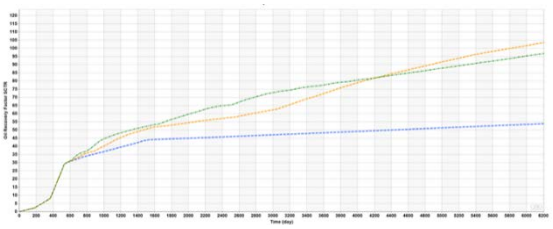


Рис. 1. Изменение коэффициента извлечения нефти во времени для вариантов с закачкой азота – вар.1 (синяя кривая), закачкой сухого воздуха вар.2 (желтая кривая), закачка смеси с ВВО 0.001 (зеленая кривая)

Сравнивая варианты с закачкой инертного газа, сухого воздуха и с ВВО (рисунок 1) видно, что рост накопленной добычи при закачке азота и сухого воздуха практически прекращается после 5 лет работы, и затем для случая с СВ через два года восстанавливается, благодаря росту отборов сырья из недренируемых интервалов пласта и получением в том числе и синтетической нефти.

Сравнение газового фактора (Гф) и температуры (Т) по вариантам с закачкой азота, сухого воздуха и водовоздушной смеси с ВВО 0.001 представлено на рисунках 2 и 3.

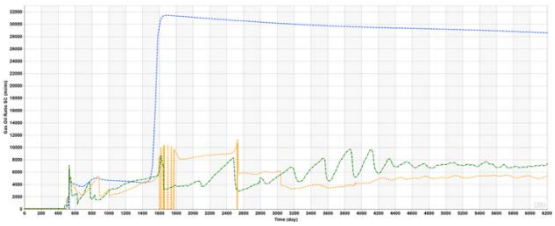


Рис. 2. Изменение газового фактора во времени для вариантов с закачкой азота – вар.1 (синяя кривая), закачкой сухого воздуха вар.2 (желтая кривая), закачка смеси с ВВО 0.001 (зеленая кривая)

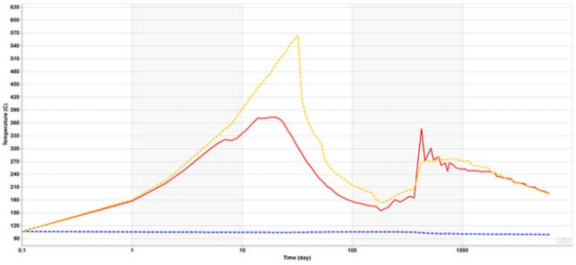


Рис. 3. Изменение температуры во времени для призабойной зоны нагнетательной скважины (расстоянии 1м от нагнетательной скважины) с закачкой азота – вар.1 (синяя кривая), закачкой сухого воздуха вар.2 (желтая кривая), закачка смеси с ВВО 0.001 (зеленая кривая)

Изменение Гф во времени для варианта 2 (закачка сухого воздуха) показало увеличение Гф до 10 тыс. $м^3/м^3$ на 5-й год разработки, сохранение данного значения в течении 3-х лет. В тоже время вариант 3 (закачка смеси с ВВО) позволяет продлить добычу нефти и УВ сырья на весь рассмотренный период 18 лет.

В целях учета технико-экономических критериев прекращение работы скважины при дальнейшем анализе принималось, что минимальный дебит по нефти составляет 1 т/сут, максимальный Гф - 10 тыс. $нм^3/м^3$, максимальное обводнение продукции – 99%.

Надо отметить, что для варианта 2 снижение Гф (на седьмой год разработки) (после более двух лет работы с аномально высокими значениями Гф) обусловлено активным вовлечением в разработку недренируемых интервалов пласта и получением углеводородного сырья в виде синтетической нефти и УВ, содержащихся в изначально (до проведения ТГВ) закрытых пустотах недренируемой породы.

Основные результаты расчетов вариантов с закачкой различных агентов и с изменением ВВО и циклов закачки приведены в таблице 2.

Таблица 2. Основные показатели расчетных вариантов при закачке воздуха и водовоздушной смеси

№ варианта	ВВО, д.ед. (н.у)	Рентабельный срок разработки, лет	Нак. добыча УВ за рент.период (н.у), $м^3$	Нак. добыча газа за рентабел. период (н.у), $м^3$	Обводнение продукции (год), %	Гф, $м^3/м^3$	Нак. воздухонефт. отношение, $м^3/м^3$	Прирост добычи УВ сравн. с вар.2 (СГ), %
1	2	3	4	5	6	7	8	9
1 (закачка N2)	0	5	7,07	15459	5,4	31335	2138	-17%
2 (сухое горение – базовый вар-т)	0	5	8,29	15806	1,6	8352	1828	0%
3	0.001	16	15,11	50132	32,3	7716	2109	45%
4	0.0015	18	16,06	56934	36,6	7400	2035	48%
5	0.002	13	14,32	43240	45,2	9325	1988	42%
6	0.005	12	13,99	39322	53,2	10385	2161	41%
7	0.0015	15	14,76	46995	31,7	7620	2097	44%
8	0.0015	13	14,33	43206	45,0	8800	1974	42%
9	0.0015	11	13,27	36330	44,6	8427	1953	38%
10 (ВВО + 30% О2)	0.0015	18	19,91	56121	22,8	4282	1780	58%
11 (ВВО + 50% О2)	0.0015	18	28,85	54031	22,2	4180	1422	71%

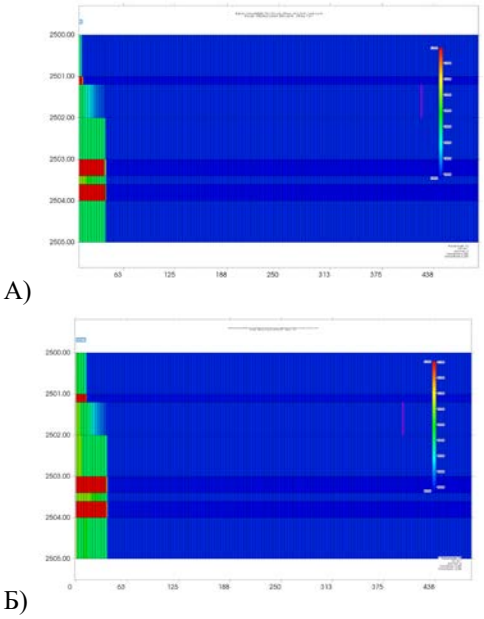
Рентабельный срок разработки для каждого варианта определялся по достижению принятых ограничений на скважину по минимальному дебиту по нефти, составившему 1 т/сут.

Величины изменения температуры в пласте (рисунок 3) показывают возможность увеличения объёма подключаемых к выработке ресурсов УВ сырья из недренируемых интервалов пласта. Увеличение температуры микропустотного пространства (изначально недренируемого) до 350ºС и выше, позволяет не только извлечь не менее 70-80% первоначально содержащихся в ней УВ, но и в результате пиролиза и крекинга превратить значительную часть твердого керогена в жидкие и газообразные углеводороды [8].

Из рисунка 3 видно, что при сухом горении (вариант 2) начальная температура возле забоя нагнетательной скважины достигает 550ºС, в то время как при закачке смеси с ВВО – 360ºС. Однако в дальнейшем фронт горения перемещается от забоя скважины и только через год температура вновь повышается, благодаря притоку УВ сырья из близлежащих недренируемых зон (матрицы породы), в которых благодаря прогреву формируются открытые поры и начинаются реакции пиролиза керогена.

На рисунке 4 (а, б) приведено изменении параметра эффективной пористости в разрезе пласта при продвижения температурного фронта (через 7 лет после начала закачки). Приведено сравнение вариантов при закачке сухого воздуха (рисунок 4а) и при водовоздушной смеси с ВВО 0.001 (рисунок 4б).

Видно, что закачка смеси с ВВО приводит к увеличению охвата пласта, включая недренируемые зоны изначально непроницаемой матрицы породы, также меняется и характер продвижения теплового потока по разрезу. Параметр эффективной пористости наиболее наглядно показывает подключение к разработке ранее недренируемых толщин в недренируемых зонах пласта за счет прогрева породы и пиролиза керогена.



А)

Б)

Рис.4. Изменение эффективной пористости при проведении ТГВ через 7 лет с начала закачки для: а) Вариант 2 (сухое горение) б) Вариант 4 (закачка водовоздушной смеси с ВВО 0.0015)

2.2. Оценка технологических показателей вариантов при проведении ТГВ с закачкой смеси с различными водовоздушными отношениями

Для определения оптимального, с точки зрения технологической эффективности процесса, водовоздушного отношения, были проведены расчеты вариантов по влажному внутрипластовому горению (ВВГ) с различными водовоздушными отношениями. Расчеты включали следующие варианты:

- ВВО 0.001 (вариант 3, соответствует максимальному, из рассматриваемых, газосодержанию (ГС) в закачиваемой смеси, составившему - 75%);
- ВВО - 0.0015 (вариант 4, ГС - 60%);
- ВВО - 0,002 (вариант 5, ГС - 67%);
- ВВО - 0.005 (вариант 6, ГС - 38%).

Компонентный состав задавался в виде объемных долей каждого компонента смеси. Рассчитанные варианты сравнивались с вариантом с сухой закачкой воздуха в одинаковых условиях.

Как отмечено выше, одной из важных задач при разработке кероген-содержащих отложений БС является расширение зон протекание процесса по площади и увеличение охвата воздействием, которое возможно при закачке термогазовых смесей с определенным водовоздушным отношением.

На рисунках 5-7 показано изменение показателей разработки по вариантам с закачкой смеси с различными водовоздушными отношениями, включая сравнение газового фактора (рисунок 5), обводнения продукции скважины (рисунок 6) и температуры (рисунки 7,8)

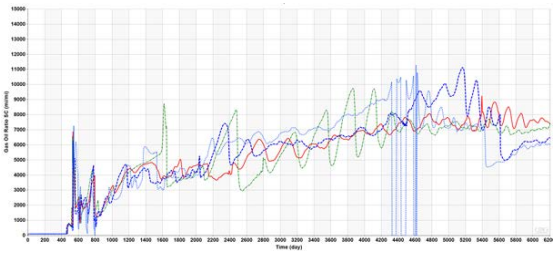


Рис. 5. Изменение газового фактора во времени для вариантов с ВВО 0,001 (вар.3 зеленая кривая), ВВО 0,0015 (вар.4 красная кривая), ВВО 0,002 (вар.5 синяя кривая) и ВВО 0,005 (вар.6 голубая кривая)

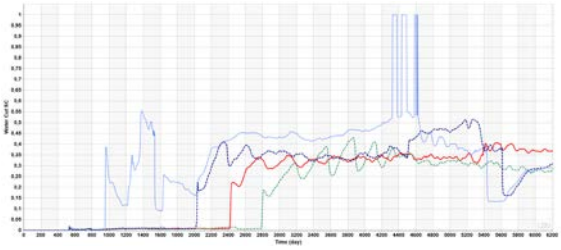


Рис. 6. Изменение обводнения продукции скважины во времени для вариантов с ВВО 0,001 (вар.3 зеленая кривая), ВВО 0,0015 (вар.4 красная кривая), ВВО 0,002 (вар.5 синяя кривая) и ВВО 0,005 (вар.6 голубая кривая)

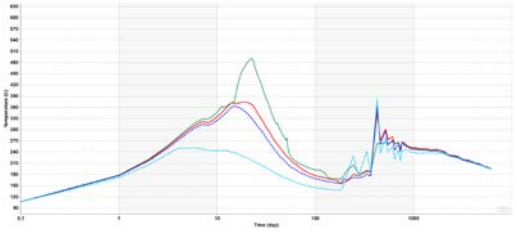


Рис. 7. Изменение температуры во времени (в log масштабе) для призабойной зоны нагнетательной скважины (расстоянии 1м от нагнетательной скважины) для вариантов с ВВО 0,001 (вар.3 зеленая кривая), ВВО 0,0015 (вар.4 красная кривая), ВВО 0,002 (вар.5 синяя кривая) и ВВО 0,005 (вар.6 голубая кривая)

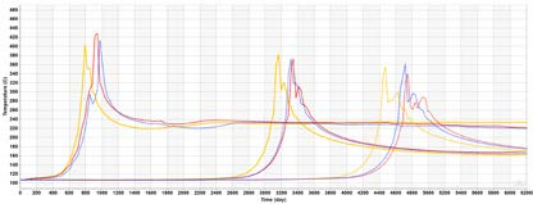


Рис.8. Изменение температуры во времени для различных зон пласта (на расстоянии 10, 50 и 70м от нагнетательной скважины) для вариантов с закачкой закачка смеси с ВВО 0,0015 (вар.4- красная кривая), закачка смеси с ВВО 0.005 - вар.6 (синяя кривая), закачка сухого воздуха – вар.2 (желтая кривая)

Изменение температуры во времени, приведено на рисунке 8 для различных зон пласта (находящихся на расстоянии 10, 50 и 70 м от нагнетательной скважины) для вариантов с закачкой закачка смеси с ВВО 0,0015 (вар.4), с ВВО 0.005 (вар.6) и закачкой сухого воздуха (вар.2). Анализ изменения температуры показывает, что с ВВО 0.0015 время прохождение теплового фронта через зоны пласта, находящиеся на расстоянии 10 и 50 м от нагнетательной скважины отстает от соответствующего времени при сухом горении (вариант 2) на 6-7 месяцев. Соответствующее отставание теплового фронта по времени от СГ на расстоянии 70 м от нагнетательной скважины, уже составляет более 9 месяцев.

Для отражения характера изменения объёмов и продвижения сформировавшейся синтетической нефти (Sint_oil) при закачке смеси с ВВО на рисунке 9 приведен разрез пласта с изменением содержания синтетики через 2 года с начала закачки. Видно, что вся образовавшаяся нефть распределяется в дренируемых пропластках и фильтруется с чередующимся изменением концентрации в общей продукции.

Для определения оптимального объёма оторочки воздуха и воды в смеси была построена зависимость накопленной добычи нефти для различных вариантов от объёма оторочки (переведенного в поровый объём пласта (PV) участка воздействия). На рисунке 10 показана данная зависимость, полученная за период семь лет, который определяет срок полного выноса агентов оторочки (в том числе воды). В дальнейшем проходит в основном отбор из недренируемых зон за счет теплового воздействия при ТГВ.

Как видно на рисунке 10 оптимальный объём оторочки составил 0,33-0,35 порового объёма пласта, что хорошо коррелируется с результатами ранее проведенных исследований [5].

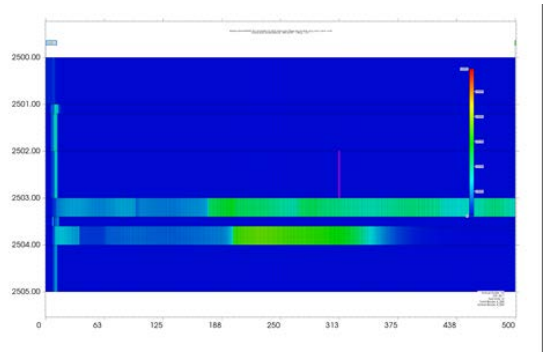


Рис.9. Изменение содержания синтетической нефти по разрезу при закачке водовоздушной смеси (вариант 4 ВВО 0.0015) через 2 года с начала закачки

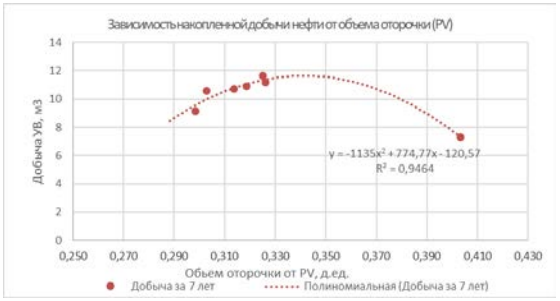


Рис.10. Изменение накопленной добычи нефти при проведении ТГВ от объёма закачки агентов (пересчитанный в поровом объёме участка воздействия)

Расчеты показали, что с повышением водовоздушного отношения температура по направлению продвижения фронта горения несколько снижается. Это происходит за счет увеличения переносимого водяным паром тепла из зоны за фронтом горения в зону перед фронтом горения и некоторым снижением общей температуры. В случае если температура в матрице падает ниже температуры начала пиролиза керогена (около 350°С), что получено для варианта 6 с ВВО 0,005, то интенсивность данной реакции снижается, вплоть до ее прекращения.

Сравнительная оценка вариантов с различ-

ными ВВО (таблица 2) показала, что для вариантов 3-5 (ВВО от 0,001 до 0,002) имеем наилучший прирост добычи по сравнению с закачкой сухого воздуха. С учетом более равномерного изменения газового фактора и обводненности продукции вариант 4 (ВВО 0,0015) был выбран в качестве оптимального.

Далее в целях выравнивания и поддержания фронта горения и равномерного охвата недренируемых зон были рассчитаны варианты с регулированием процесса и изменением соотношения и продолжительности циклов закачиваемых агентов.

2.3. Оценка технологических показателей вариантов при проведении ТГВ с закачкой смеси с различными продолжительностью и периодичностью циклов закачки

Третий этап включал расчеты и оценку вариантов (варианты №№ 4, 8, 9) при закачке смеси различными циклами и периодами закачки агентов с оптимальным водовоздушным отношением, определенным по результатам расчета второго этапа.

Во всех вариантах соответствующим этапу 3 проводится циклическая закачка сухого воздуха и водовоздушной смеси с ВВО 0,0015 продолжительностью два года, (до создания оторочки с объемом агентов около 32-33 % порового объема участка воздействия), затем переход на закачку сухого воздуха. Первые два года идет чередование закачки водовоздушной смеси и сухого воздуха с различными периодами.

По варианту 4 цикл включает закачку 2месяца сухого воздуха (СВ) + 1 месяц смесь ВВО (всего 9 циклов закачки), вариант 8 – 1 месяц СВ + 1 месяц ВВО (12 циклов закачки), варианту 9 – 1 месяц (СВ) + 2 месяца ВВО (всего 9 циклов закачки.

На рисунках 11-13 показано изменение показателей разработки по вариантам с закачкой смеси с различными циклами, включая сравнение обводнения продукции скважины (рисунок 11), газового фактора (рисунок 12), и температуры (рисунок 13).

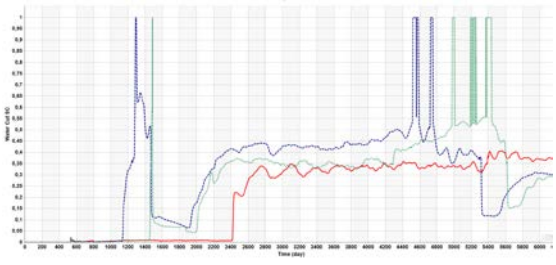


Рис. 11. Изменение обводнения продукции скважины во времени для вариантов 4 (красная кривая), варианта 8 (зеленая кривая) и варианта 9 (синяя кривая)

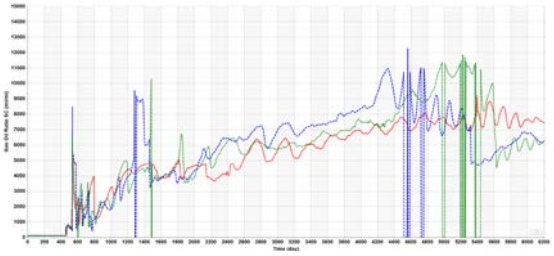


Рис. 12. Изменение газового фактора во времени для вариантов 4 (красная кривая), варианта 8 (зеленая кривая) и варианта 9 (синяя кривая)

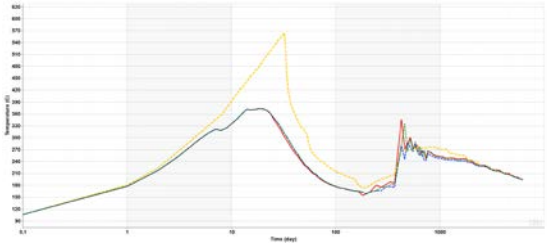


Рис. 13. Изменение температуры (в log масштабе) для призабойной зоны нагнетательной скважины (расстоянии 1м от нагнетательной скважины) во времени для вариантов 2 (СВ –желтая кривая), 4 (красная кривая), варианта 8 (зеленая кривая) и варианта 9 (синяя кривая)

Как видно на рисунках 11 и 12 только вариант 4 с циклом закачки – 2 месяца СВ и 1 месяц ВВО показывает стабильную работу добывающей скважины в течении всего периода расчета. Изменение температуры в ближайшей к стволу скважины зоне (на расстоянии 1 м) практически совпадает во всех рассмотренных вариантах.

В связи с тем, что данный вариант 4 имеет и максимальный прирост добычи УВ (таблица 2) можно считать вариант 4, имеющим наилучшие технологические показатели, оптимальным выбором при проведениии ТГВ для принятых геолого-физических условий.

2.4. Оценка технологических показателей вариантов при проведении ТГВ с закачкой обогащенной смеси

В настоящем разделе приведены результаты расчетов вариантов с закачкой обогащенной смеси (содержание кислорода в воздухе повышено до 30% (вариант 10) и до 50% (вариант 11).

Для повышения эффективности проведения ТГВ возможно проведение закачки воздуха с увеличением концентрации кислорода. Это влияет на скорость реакции, зависящей от концентраций реагирующих веществ и на скорость тепловыделения, которая экспоненциально увеличивается с температурой.

Задача применения обогащенного кислородом воздуха в настоящее время может быть реализована, но характеризуется повышенной стоимостью при технологическом обустройстве

участка воздействия.

На рисунках 14-17 показано сравнение показателей разработки по вариантам с закачкой сухого воздуха (вариант 2), смеси с ВВО (вариант 4), обогащенной смеси (кислорода в воздухе 30% (вариант 10) и 50% (вариант 11), включая сравнение обводнения продукции скважины (рисунок 14), газового фактора (рисунок 15), и температуры (рисунки 16 и 17).

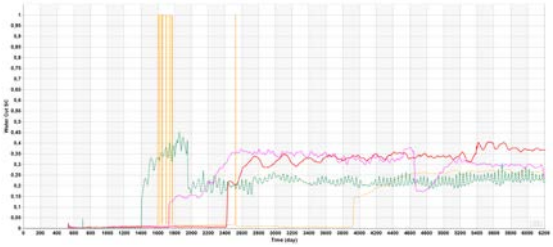


Рис. 14. Изменение обводнения продукции скважины во времени для вариантов: закачки СВ (вар.2 –желтая кривая), смеси с ВВО (вар.4 - красная кривая), варианта 8 (30% О2 -малиновая кривая) и варианта 9 (50% О2 - зеленая кривая)

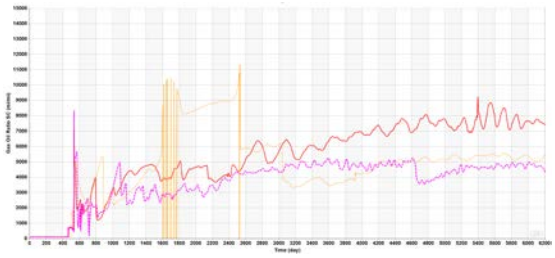


Рис. 15. Изменение газового фактора во времени для вариантов: закачки СВ (вар.2 –желтая кривая), смеси с ВВО (вар.4 - красная кривая), варианта 8 (30% О2 - малиновая кривая) и варианта 9 (50% О2 - зеленая кривая)

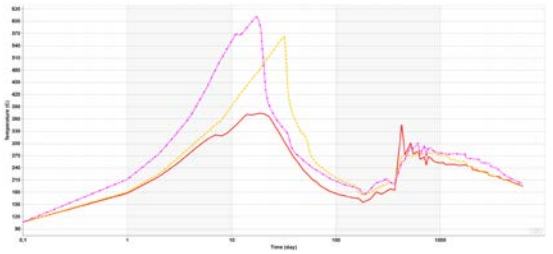


Рис. 16. Изменение температуры (в log масштабе) для призабойной зоны нагнетательной скважины (расстоянии 1м от нагнетательной скважины) во времени для вариантов закачки СВ (вар.2 –желтая кривая), смеси с ВВО (вар.4 - красная кривая), варианта 8 (30% О2 -малиновая кривая) и варианта 9 (50% О2 - зеленая кривая)

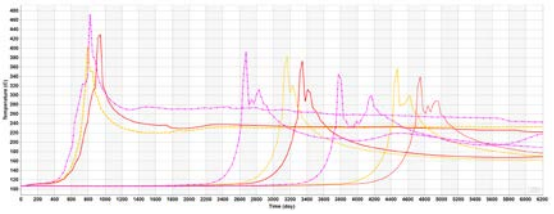


Рис. 17. Изменение температуры во времени для различных зон пласта (на расстоянии 10, 50 и 70м от нагнетательной скважины) для вариантов закачки СВ (вар.2 –желтая кривая), смеси с ВВО (вар.4 - красная кривая), варианта 10 (30% О2 -малиновая кривая)

На рисунке 17, где приведена динамика продвижения температурного фронта по простиранию пласта видно, что время продвижения фронта горения (на расстоянии уже 70м) для варианта 8 (30% О2) опережает аналогичный фронт при ВВО 0.0015 (вариант 4) почти на 3 года, а фронт при закачке СВ (вариант 2) – на 2 года.

Использование обогащенной смеси при ТГВ безусловно приводит к ускорению продвижения фронта горения и увеличению охвата пласта, включая недренируемые зоны изначально непроницаемой матрицы породы. В результате ускоренного продвижения фронта происходит и увеличение охвата по пласту, которое составляет до 20% по сравнению с сухим горением.

2.5 Оценка дополнительной добычи синтетической нефти и УВ сырья (нативной нефти) из недренируемых интервалов пласта

В процессе работы проведена количественная оценка таких важных составляющих накопленной добычи, как дополнительная добыча синтетической нефти (в стандартных условиях) из дренируемой и недренируемых зон и добыча УВ сырья (нативной нефти) из недренируемых (до проведения ТГВ) закрытых пустот в недренируемых интервалах (зонах) пласта.

На рисунке 18 показано изменение доли добываемых углеводородных компонентов нефти во времени по варианту 4 (с ВВО 0.0015), включая отбор С5-С7 (коричневая кривая), С8-С11 (коричневая кривая - пунктир), синтетической нефти, сформировавшейся в матрице (красная кривая). Также на рисунке 18 показан отбор синтетической нефти, по варианту 11(закачка 50% О2 с ВВО 0,0015), сформировавшейся только в матрице (сплошная малиновая кривая) и синтетической нефти (вариант 9), сформировавшейся в дренируемых интервалах (пунктир, малиновая кривая).

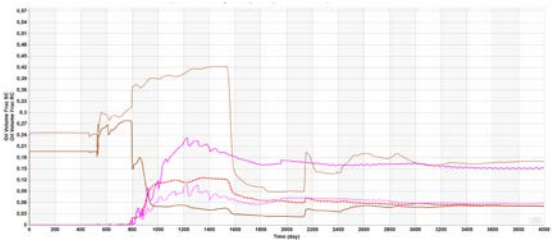


Рис. 18. Изменение доли добываемых УВ компонентов во времени по варианту 4, включая отбор С5-С7 (коричневая кривая), отбор С8-С11 (коричневая кривая - пунктир), отбор синтетической нефти из матрици (красная кривая), для варианта 11 - отбор синтетической нефти из матрицы (сплошная малиновая кривая) и синтетической нефти из дренируемых интервалов (пунктир, малиновая кривая)

Из рисунка 18 видно, что доля синтетической нефти, сформировавшейся в матрице по варианту 4 (ВВО 0,0015) достигает максимального значения в 12 % от всей добываемой УВ продукции, и затем стабилизируется на уровне 6%.

Для варианта 11 (50% О2) доля синтетической нефти, сформировавшейся в матрице значительно увеличивается и достигает максимального значения в 22 % от всей добываемой УВ продукции, и затем стабилизируется на уровне 16%.

Таким образом для максимального вовлечения запасов и ресурсов из недренируемых зон пласта баженовской свиты можно рекомендовать проведение работ с закачкой воздуха с увеличенной концентрацией кислорода с целью расширения объёма воздействия на недренируемые интервалы породы и более полного преобразования органического вещества баженовской свиты в жидкие и газообразные углеводороды.

Сравнение эффективности применения различных методов при проведении ТГВ показана на рисунке 19.

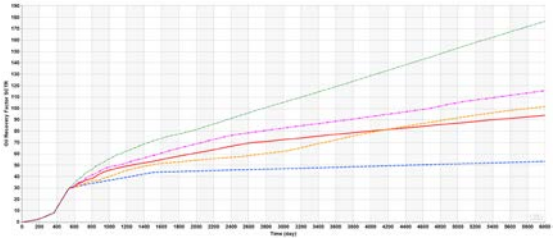


Рис. 19. Изменение коэффициента извлечения нефти во времени для вариантов с закачкой азота – вар.1 (синяя кривая), закачкой сухого воздуха вар.2 (желтая кривая), закачка смеси с ВВО 0.0015 (красная кривая), варианта 10 (30% О2 -малиновая кривая), варианта 11 (50% О2 –зеленая кривая)

Из рисунка 19 видно, что увеличение добычи УВ по варианту 11 (50% О2) по сравнению с закачкой сухого воздуха составит более 70 %. При этом основной прирост добычи получен за счет притока УВ сырья из подключаемых интервалов ранее недренируемых керогенсодержащих зон пласта.

4. Заключение

Проведенный выше анализ позволяет сделать следующие выводы и заключения:

- Проведенные гидродинамические расчеты с использованием неизотермической композиционной модели пласта для отложений баженовской свиты с изменением состава вытесняющего агента, циклов и периодов закачки и других параметров показали, что для оптимизации извлечения всего комплекса углеводородных соединений из дренируемых и недренируемых интервалов и зон пласта БС важно подбирать и реализовывать оптимальные технологические решения по выбору значений водовоздушного отношения, продолжительности циклов и режимов работы скважин;
- Оценка различных режимов закачки агентов при проведении ТГВ показала, что для обеспечение эффективного извлечения нефти из недренируемых зон желательно использовать смеси с водовоздушным отношением от 0,001-0,002;
- Расчеты показали, что использование керогена в качестве топлива для самопроизвольных окислительных процессов позволяет поддерживать высокую температуру на фронте горения и были определены оптимальные соотношения вытесняющих агентов для увеличения охвата как дренируемых, так и подключаемых к разработке непроницаемых зон баженовской свиты;
- Согласно приведенному сопоставлению дополнительный эффект от реализации влажного горения составил около 10% в добыче жидких УВ. Наиболее предпочтительным выглядит вариант с ВВО 0.0015, так как при его реализации более равномерно поддерживается процесс горения в матрице. Это позволяет наиболее полно реализовать потенциал термогазового метода и получить дополнительную добычу в виде синтетической нефти при пиролизе керогена.
- Проведенный расчет показал, что при использовании агентов с повышенным содержанием кислорода для воздействия на баженовскую свиту получено значительное увеличение технологической эффективности;
- Закачка воздуха с повышенным содержанием кислорода, будет способствовать повышению эффективности термогазового воздействия

за счёт расширения объёма воздействия на недренируемые интервалы породы и более полного преобразования органического вещества баженовской свиты в жидкие и газообразные углеводороды;

• Также необходимо проведение постоянного контроля за процессом закачки воздуха методами промысловых геофизических и гидродинамических исследований, одновременно с расширением экспериментальных и численных исследований при разработке отложений данного типа, имеющих большой ресурсный потенциал.

Работа выполнена при поддержке гранта РФФИ № 18–07-00504_А.

Water-Air Mixture Injection Efficiency in Thermal-Gas Oil Recovery Method for Bazhenov Suite Reservoir

D.T. Mironov, A.A. Glushakov, P.V. Yalov

Abstract. Problems of three-phase flow mathematical modeling for thermo-gas enhanced oil recovery method with water-air mixture injection - are considered. Simulation results of the different scenarios with water-air mixture injection in bazhenov kerogen content rock types and some parameters of termo-gas method are discussed. The main features of the applied wet in-situ combustion technology discussed and most effective scenarios are suggested.

Keywords: thermal-gas treatment, EOR methods, bazhenov formation, oxygen-containing mixture injection, kerogen, miscible drive, wet in-situ combustion, enhanced oil recovery.

Литература

1. В.Ю. Алекперов, В.И. Грайфер, Н.М. Николаев, В.Б. Карпов, В.И. Кокорев, Р.Г. Нургалиев, А.П. Палий, А.А. Боксерман, В.А. Клинчев, А.В. Фомкин, Новый отечественный способ разработки месторождений баженовской свиты (часть 1). «Нефтяное хозяйство» № 12. 2013, 100–105.

2. Д.Т. Миронов, П.В. Ялов. Оценка вовлечения в разработку недренируемых керогенсодержащих зон баженовской свиты при моделировании термогазового воздействии. «Труды НИИСИ РАН» том 9 (2019), № 6, 94 -103.

3. Д.Т. Миронов, А.А. Глушаков, П.В. Ялов. Влияние закачки водовоздушной смеси на эффективность термогазового воздействия для баженовской свиты при 2D моделировании. «Труды НИИСИ РАН» том 9 (2019), № 6, 104 -110.

4. А.А. Боксерман. Результаты и перспективы применения тепловых методов воздействия на пласт // Тепловые методы воздействия на пласт (Материалы межотраслевого семинара, г. Ухта, 5-8 октября 1971 г.), ВНИИОЭНГ, Москва, с. 10-16.

5. А.А. Боксерман, А.А. Кочешков. А.Г. Тарасов. Совершенствование технологий тепловых методов повышения нефтеотдачи нефтяных месторождений. В сб. Итоги науки и техники «ВИНИТИ» Том 24, 1993, 3-81.

6. Д.Т. Миронов, С.Г. Вольпин, В.А. Юдин. Технологические подходы при эксплуатации скважин баженовской свиты и оценка возможности подключения в разработку ресурсов недренируемых зон. «Вестник Кибернетики» 2018. № 3 С. 233–246.

7. Д.Т. Миронов, П.В. Ялов. Оценка влияния гистерезиса относительных фазовых проницаемостей при термогазовом воздействии с закачкой водовоздушной смеси. «Труды НИИСИ РАН» том 10 (2020), № 6.

8. В.И. Кокорев. Технико-технологические основы инновационных методов разработки месторождений с трудноизвлекаемыми и нетрадиционными запасами нефти / Автореф. дис. д-ра техн. наук. М.: ОАО «Российская инновационная топливно-энергетическая компания» (РИТЭК), 2010. - 58 с.

Оценка влияния гистерезиса относительных фазовых проницаемостей при проведении термогазового воздействия с закачкой водовоздушной смеси

Д.Т. Миронов[1], П.В. Ялов[2]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, mdt1958@yandex.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, petryalov@gmail.com

Аннотация. Рассматриваются вопросы математического моделирования трехфазной фильтрации в условиях применения термогазового воздействия на пласт (ТГВ) с циклической закачкой водовоздушных смесей с различным соотношением вытесняющих агентов. Расчеты на 2D линейной модели с наличием недренируемых интервалов и со средними фильтрационно-емкостными и геохимическими параметрами, соответствующими показателям для баженовской свиты, показали влияние механизма гистерезиса ОФП на эффективность проведения ТГВ.

Ключевые слова: МУН, баженовская свита, кероген, термогазовое воздействие, водовоздушное отношение, защемленный газ, гистерезис, уравнение Лэнда.

1. Введение

В процессе термогазового метода увеличения нефтеотдачи, важной составляющей, направленной на расширение охвата недренируемых интервалов пласта воздействием, является закачка термогазовых смесей с определенным водовоздушным отношением (ВВО) [1]. В этом процессе к эффекту подключения недренируемых зон (матрицы породы) добавляется и расширение зоны прогрева по толщине пласта за счет переноса тепла водяным паром.

В случае закачки водовоздушных смесей необходимо привести оценку влияния гистерезиса кривых относительных фазовых проницаемостей (ОФП) для несмачивающихся фаз на эффективность моделирования термогазового воздействия (ТГВ) на пласт.

Расчеты выполнялись на 2D линейной модели с учетом подключения недренируемых интервалов (матрицы) в отложениях баженовской свиты при проведении ТГВ, детально рассмотренной в предыдущих статьях [2, 3]. Результаты полученные в настоящей статье будут использованы при проведении дальнейшей оценки применения различных модификаций ТГВ, направленных на повышение эффективности разработки керогенсодержащих отложений баженовской свиты.

2. Особенности моделей гистерезиса ОФП при термогазовом воздействии

Метод термогазового воздействия на пласт соединяет в себе элементы тепловых и газовых методов воздействия и основная его эффективность определяется увеличением степени охвата пласта вытеснением при закачке в пласт термогазовой смеси, за счет внутрипластовой генерации высокоэффективного вытесняющего газового агента (вытесняющей оторочки), обеспечивающего значительный прирост нефтеотдачи

Результаты многочисленных исследований трехфазной фильтрации в пористой среде проведенные как на керновом материале, так и при моделировании процесса водогазового воздействия и подтвержденные исторически полученными данными, в процессе применения различных методов увеличения нефтеотдачи (МУН), показали необходимость учета механизма гистерезиса в гидродинамических моделях.

Характер и параметры кривых фазовых проницаемостей и капиллярного давления при трехфазной фильтрации зависят от насыщенности, от истории изменении насыщенности при разработке залежи (дренаж - пропитка) и также от направления ее изменения.

Данный механизм, определяемый как гистерезис ОФП и капиллярного давления (P_c), определяется следующими основными причинами - эффектом защемления вытесняемой фазы, разницей величины контактного угла смачивания

породы в циклах впитывания и дренажа, и изменением смачиваемости породы при разработке и применении МУН.

Таким образом при применении чередующихся закачек воды и воздуха в процессе проведения ТГВ необходим учет в том числе изменения кривой относительной проницаемости для циклов дренажа и пропитки, и учёт объёмов защемленного газа (как несмачивающейся фазы в системе газ-жидкость).

Процесс вытеснения для каждой из фаз и, соответственно, эффекты при гистерезисе зависят от распределения флюидов в порах, которое определяется типами породы по смачиваемости. Кроме зависимости и влияния на эффекты смачиваемости породы, гистерезис сильно влияет на процессы закачки агентов при изменении циклов, при изменении величины поверхностного натяжения и капиллярного давления.

Для описания эффектов при гистерезисе существуют различные эмпирические модели, описывающие процесс гистерезиса для двухфазных систем нефть-вода и газ-нефть (для обеих фаз - смачивающейся и несмачивающейся) и включающие механизм защемления несмачивающейся фазы и снижения фазовой проницаемости при обратимости процесса.

Модели описания гистерезиса, например модели Карлсона и Киллоу [4], включают прогноз защемления несмачивающейся фазы, снижение фазовой проницаемости при процессе пропитки и используют одни и те же сканирующие кривые при переходах от дренажа к пропитке и обратно. В методе Киллоу (см. рис. 1), реализованном в ряде программных продуктов, в том числе в ПО «Eclipse» [5], расчет защемленной насыщенности несмачивающейся фазы (S_{ncrt}), масштабируется по концевым значениям и определяется формулам (1), (2). Фазовая проницаемость на сканирующей кривой $K_{rn}(S_n)$ определяется по формулам (3), (4).

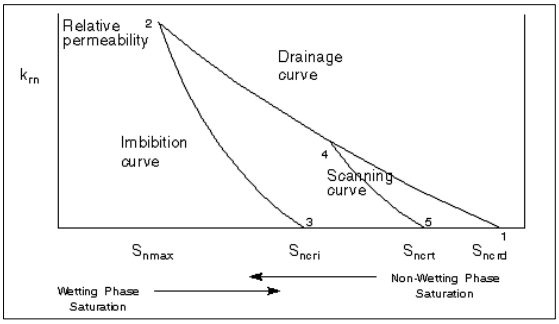


Рис. 1. Типовая модель гистерезиса несмачивающейся фазы [5]

* цифры 1-5 соответствует шагам расчета при переходе от дренажа к пропитке

$$S_{ncrt} = S_{ncrd} + \frac{S_{hy} - S_{ncrd}}{1 + C(S_{hy} - S_{ncrd})} \quad (1)$$

$$C = \left(\frac{1}{S_{ncri} - S_{ncrd}}\right) - \left(\frac{1}{S_{n\,max} - S_{ncrd}}\right) \quad (2)$$

$$K_{rn}(S_n) = \frac{K_{rni}(S_{norm})K(S_{hy})}{K_{rnd}(S_{n\,max})} \quad (3)$$

$$S_{norm} = \frac{S_{ncri} + (S_n - S_{ncrt})(S_{n\,max} - S_{ncri})}{S_{hy} - S_{ncrt}} \quad (4)$$

где S_{ncri} – критическая насыщенность несмачивающейся фазы при пропитке; S_{ncrd} – критическая насыщенность несмачивающейся фазы при дренаже; S_{ncrt} - значение защемленной насыщенности несмачивающейся фазы; S_{hy} - заданное значение насыщенности при гистерезисе; $K_{rni(d)}$ - величины фазовой проницаемости на кривой пропитки и дренажа; $S_{n\,max}$ - максимальное значение насыщенности несмачивающейся фазы.

В данной модели, величина защемленной насыщенности всегда находится между критическими величинами при дренаже и пропитке, кроме случая совпадения данных кривых.

2.1. Обоснование параметров модели защемленного газа при трехфазной фильтрации

Насыщенность защемленным газом значительно влияет на процессы, происходящие при мультифазных закачках, к которым относятся и процесс закачки водовоздушной смеси. От данного параметра зависит как приемистость нагнетательных скважин и характер распространения воды в пласте, так и в первую очередь количество газа, остающегося в пласте и не принимающего участия в процессе вытеснения. Данные по защемленному газу по результатам экспериментов хорошо аппроксимируются модифицированной формулой Лэнда (5) [6]:

$$S_{gt}(S_{g\,max}) = \frac{S_{g\,max}}{1 + \left(1/S_{gr} - 1\right)(S_{g\,max})^{1+\frac{bS_{gr}}{(1-S_{gr})}}} \quad (5)$$

где S_{gmax} - максимальная газонасыщенность, имеющая место в данном пласте, критическая газонасыщенность S_{gr} (S_{gt} при $S_{g\,max}$=1) и b являются эмпирически полученными константами (в уравнении Лэнда b=0). Это уравнение отражает идею о том, что максимальная газонасыщенность определяет количество защемленного газа в мультициклических процессах.

При числовом моделировании формула Лэнда используется для расчета насыщенности

свободного газа в процессе вытеснения по сканирующим кривым для оценки эффективной проницаемости для газа. По мере роста S_{gmax} при процессах дренажа и впитывания это вызывает снижение проницаемости.

По результатам опытов по вытеснению нефти пластовой водой и газом с циклами дренажа - пропитки получена зависимость защемленного газа $S_{gas\ trp}$ от максимальной газонасыщенности S_{gmax}. На рис. 2 представлен график, содержащей как экспериментальные данные, так и полученные по модифицированному уравнению Лэнда [7].

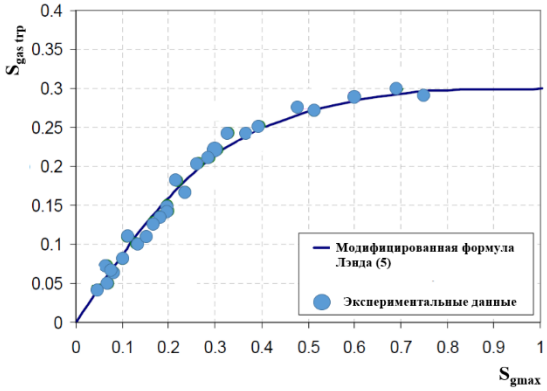


Рис. 2. Зависимость защемленного газа $S_{gas\ trp}$ от максимальной газонасыщенности S_{gmax} [7].

2.2. Модель гистерезиса ОФП реализованная в гидродинамическом симуляторе

В модели, представленной в настоящей статье, построены относительные фазовые проницаемости как для дренажа, так и пропитки с учетом дополнительных концевых точек (см. рис. 3а, б).

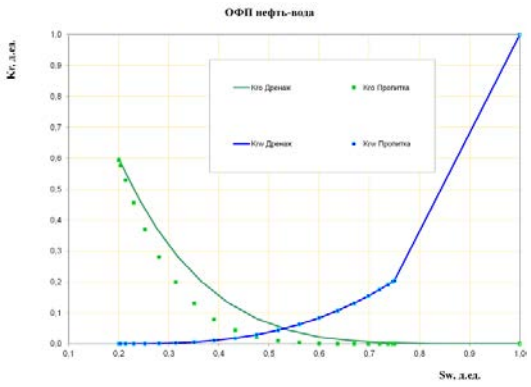


Рис. 3а. Относительные фазовые проницаемости с учетом эффекта гистерезиса для несмачивающейся фазы (дренаж – сплошная кривая, пропитка – пунктирная линия) в системе нефть-вода

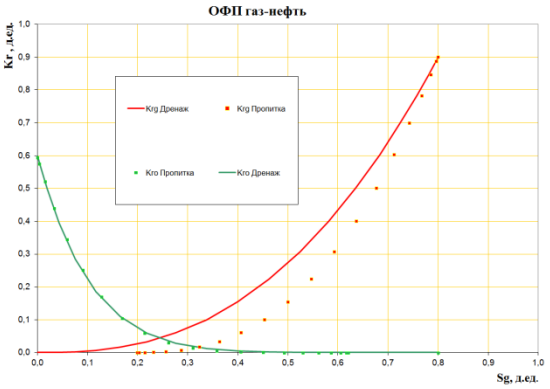


Рис. 3б. Относительные фазовые проницаемости с учетом эффекта гистерезиса для несмачивающейся фазы (дренаж – сплошная кривая, пропитка – пунктирная линия) в системе газ-нефть

При построении ОФП принимались следующие значения концевых точек – остаточная нефтенасыщенность после вытеснения водой - 25 %, остаточная нефтенасыщенность после вытеснения газом - 18 %, максимальная критическая газонасыщенность – 20 %.

3. Основные подходы, реализованные в линейной модели пласта при расчете вариантов разработки и оценке технологической эффективности применения ТГВ

Для численной реализации процесса ТГВ на гидродинамическом симуляторе STARS (CMG Канада) была подготовлена 2D модель, включающая возможность вовлечения углеводородного сырья из изначально недренируемых зон (матрицы) баженовской свиты в процессе термогазового воздействия [2].

Принципиальная схема (см. рис. 4), используемая в линейной модели, включает элемент пласта, расположенный между нагнетательной и добывающей скважинами, с недренируемыми слоями (интервалами матрицы) и со средними фильтрационно-емкостными и геохимическими параметрами характерными для данных отложений.

Пласт был представлен чередованием недренируемых кероген-содержащих непроницаемых пропластков и проницаемых интервалов толщиной до 0,4 м. Недренируемые пропластки содержат основную часть органического вещества нефтематеринской породы – керогена, а приток флюидов из скважины и закачка агентов в пласт проводится из дренируемых интервалов.

Для оценки эффектов гистерезиса ОФП на данной линейной модели, включающей матрицу и дренируемую зону, были проведены расчеты

трех вариантов:

Вариант 1 - предусматривает проведение ТГВ с закачкой сухого воздуха (СВ) в нагнетательную скважину при постоянном расходе 10 $м^3$/сут.

Вариант 2 - Для оценки влияния гистерезиса ОФП проводится циклическая закачка сухого воздуха и водовоздушной смеси с ВВО 0,0015 продолжительностью два года, (до создания оторочки с объемом агентов около 30-35 % порового объема участка воздействия), затем переход на закачку сухого воздуха на период в 16 лет. Первые два года идет чередование закачки водовоздушной смеси и сухого воздуха с периодами - 1 месяц ВВО 0,0015 и 2 месяца сухой воздух, всего 9 циклов закачки. Отбор нефти по скважине ведется при забойном давлении составляющим 0,9 от давления насыщения (16,6 МПа).

Вариант 3 - все условия закачки и отбора проводятся аналогично *варианту 2*, но без учета эффекта гистерезиса ОФП для не смачивающихся флюидов.

Исходные параметры вариантов с учетом и без учета эффектов гистерезиса ОФП при чередовании закачиваемых агентах – сухого воздуха и смеси с ВВО приведены в таблице 1.

На рис. 5 - 7 приведено сравнение технологических показателей разработки данных вариантов. На рис. 5, где показано изменение газового фактора (ГФ) во времени, видно, что в случае неучета эффекта гистерезиса ОФП (*вариант 3*) уже на 5-й год получаем значительное повышение ГФ до 18 тыс. $м^3/м^3$, совместно с неравномерным изменением данного параметра. Это превышение, при учете технико-экономических критериев, когда дебит по нефти становится менее 1 т/сут, приводит к остановке эксплуатации скважины. Для *варианта 1* (закачка сухого воздуха) ГФ достигает 10 тыс. $м^3/м^3$, что также показывает экономически нецелесообразность продолжения разработки.

Надо отметить, что дальнейшее снижение ГФ (на седьмой год разработки) и обводненности продукции после полутора лет работы с аномально высокими значениями ГФ обусловлено вовлечением в разработку недренируемых интервалов (матрицы) пласта и получением углеводородного сырья в виде синтетической нефти и УВ, содержащихся в изначально (до проведения ТГВ) закрытых пустотах недренируемой породы.

Из рис. 7 видно, что в варианте без учета гистерезиса ОФП добыча УВ на 10% меньше, чем в варианте с гистерезисом в период первых 5 лет (до окончания экономически рентабельного периода разработки). Это объясняется следующими основными причинами:

• Без учета эффекта гистерезиса происходит преждевременный прорыв газа (прежде всего азота), приводящего к отключению добывающей скважины.

• В варианте с гистерезисом значительный рост ГФ не происходит за счет того, что при переходе с закачки водогазовой смеси на цикл закачки воздуха, относительная фазовая проницаемость для газа резко уменьшается и требуется повторный рост насыщенности для соответствующей несмачивающейся фазы.

• Также, наличие высоких значений защемленного газа (в среднем - 10%, максимум - 20%) в порах коллектора, который остается неподвижным при возврате на цикл закачки газовой смеси, и относительная фазовая проницаемость для газа восстанавливается только после превышения критической газонасыщенности.

Результаты расчетов вариантов с учетом и без учета эффектов гистерезиса ОФП при чередовании закачиваемых агентов – сухого воздуха и смеси с ВВО приведены в таблице 2.

На рис. 8 приведено изменение газонасыщенности после 5 лет закачки водовоздушной смеси с ВВО 0,0015 для *варианта 2* (с учетом гистерезиса ОФП) и *варианта 3* (без учета гистерезиса ОФП). Видно, что зона повышенной газонасыщенности без учета гистерезиса достигает добывающий скважины значительно быстрее и имеет более высокие абсолютные значения (что соответствует высокому ГФ). Таким образом, учет механизма гистерезиса ОФП и наличие защемленного газа в коллекторе, приводит к тому, что продвижение фронта смеси происходит более равномерно, при более высоких фильтрационных сопротивлениях, и коэффициент охвата пласта более высокий.

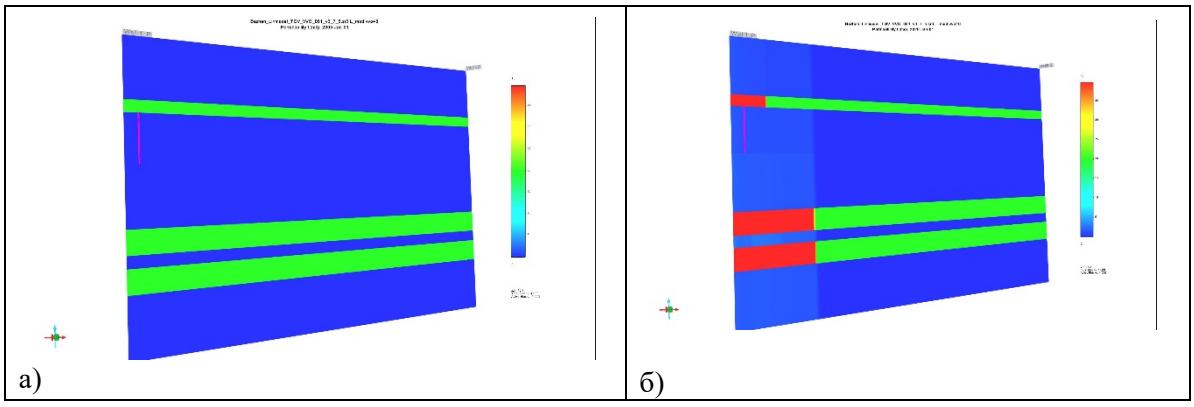


Рис. 4. Изменение проницаемости на принципиальной схеме участка воздействия при закачке сухого воздуха а) до начала закачки б) по окончанию закачки

Таблица 1. Рассмотренные варианты закачки воздуха и водовоздушной смеси

№ варианта	ВВО, д.ед. (н/у)	Газосодержание (ГС), %	Объем закачки смеси (в PV уч-ка ОПР, %	Среднее давление закачки смеси ($P_{заб}$ нагн/доб), МПа	Тип закачиваемого агента	Период и цикл закачки ТГВ
1	0	100	0,283	30,0/15,0	Воздух	Постоянная закачка
2	0,0015	67	0,319	40,0/15,0	Водо-воздушная смесь	Циклическая закачка: 2 месяца воздух + 1месяц ВВО (9 циклов)
3	0,0015	67	0,344	40,0/15,0	Водо-воздушная смесь	Циклическая закачка. 2 месяца воздух + 1месяц ВВО (9 циклов)

Таблица 2. Основные показатели расчетных вариантов при закачке воздуха и водовоздушной смеси с учетом и без учета гистерезиса ОФП

№ варианта	ВВО, д.ед. (н.у)	Рентабельный период, лет	Накопленная добыча УВ за рентабельный период (н.у), $м^3$	Накопленная добыча газа за рентабельный период (н.у), $м^3$	Обводнение продукции (год), %	Газовый фактор, $м^3/м^3$	Накопленное воздухонефтяное отношение, $м^3/м^3$	Прирост добычи УВ в сравнении с *вариантом 1* (СВ), %
1	0	5	8,29	15806	1,6	8352	1828	0%
2	0,0015	18	16,06	56934	36,6	7400	2035	48%
3	0,0015	5	9,03	15714	1,0	7511	1653	8%

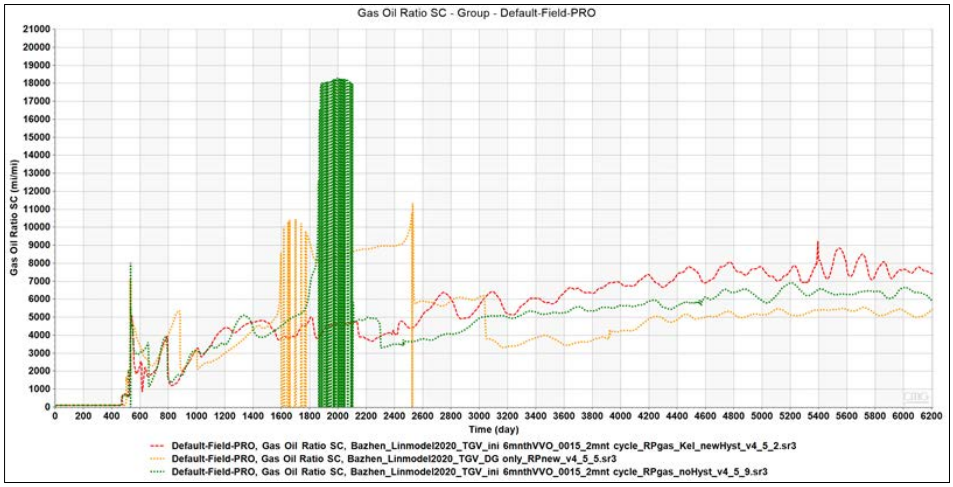


Рис. 5. Изменение газового фактора во времени для варианта 1 - закачка сухого воздуха (желтая кривая), *варианта 2* – закачка смеси с ВВО с учетом гистерезиса ОФП (красная кривая) и *варианта 3* – закачка смеси с ВВО без учета гистерезиса ОФП (зеленая кривая)

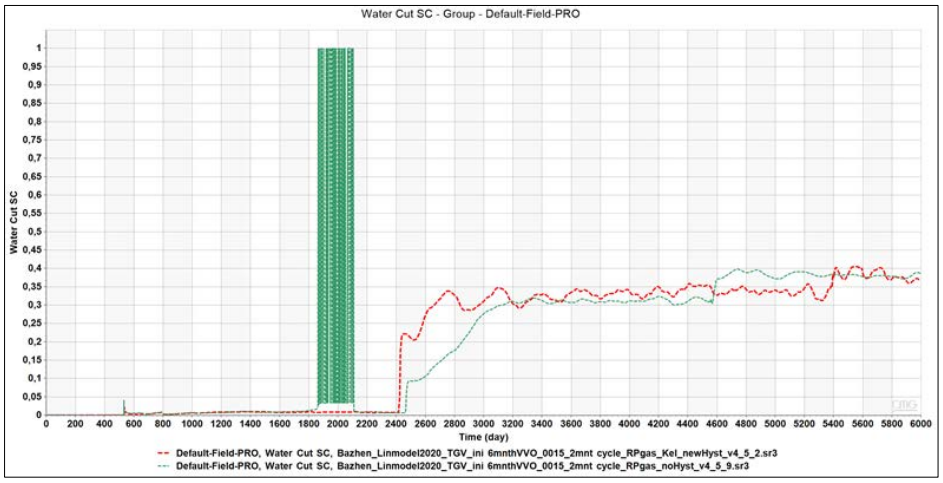


Рис. 6. Изменение обводненности продукции во времени для *варианта 2* – закачка смеси с ВВО с учетом гистерезиса ОФП (красная кривая) и *варианта 3* – закачка смеси с ВВО без учета гистерезиса ОФП (зеленая кривая)

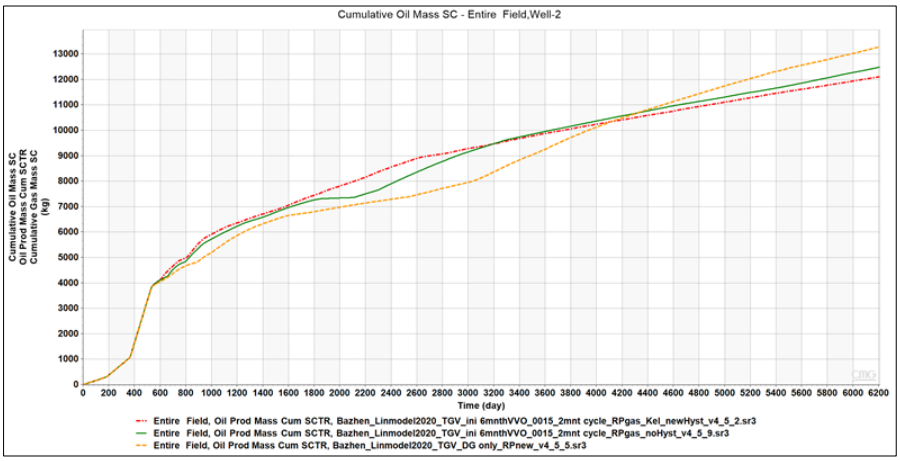


Рис. 7. Изменение накопленной добычи УВ во времени для варианта 1 - закачка сухого воздуха (желтая кривая), *варианта 2* – закачка смеси с ВВО с учетом гистерезиса ОФП (красная кривая) и *варианта 3* – закачка смеси с ВВО без учета гистерезиса ОФП (зеленая кривая

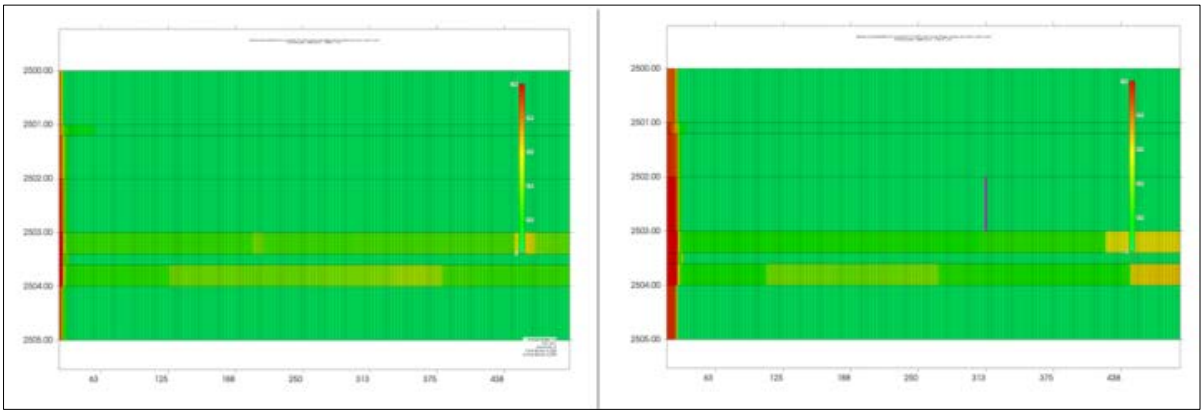


Рис. 8. Изменение газонасыщенности при ТГВ через 5 лет после начала закачки водовоздушной смеси (ВВО 0.0015) по - а) *варианту 2* – с учетом гистерезиса ОФП б) *варианту 3* – без учета гистерезиса ОФП

4. Заключение

Проведенный анализ позволяет сделать следующие выводы и заключения:

• При моделировании термогазового воздействия на пласт с циклической закачкой водовоздушных смесей с различными соотношениями вытесняющих агентов, формируется разнонаправленные изменения насыщенностей фаз.

• При проведении чередующихся закачек воды и воздуха необходим учет изменений относительной фазовой проницаемости при дренаже и пропитке (впитывании) и учёт объёмов защемленного газа (несмачивающейся фазы в системе газ-жидкость).

• Расчеты на 2D линейной модели по схеме - нагнетательная – добывающая скважина, с наличием недренируемых интервалов (подключаемых к разработке при ТГВ) со средними фильтрационно-емкостными и геохимическими параметрами, соответствующими показателям для баженовской свиты, показали влияние механизма гистерезиса ОФП на эффективность проведения ТГВ.

• Приведенные данные показывают, что учет явления гистерезиса фазовых проницаемостей, приводит к корректной величине коэффициента охвата пласта воздействием. Продвижение фронта водовоздушной смеси происходит более равномерно за счет изменения относительных фазовых проницаемостей и наличия защемленного газа в коллекторе в процессе изменения направления насыщенности (дренаж -пропитка) при разработке залежи.

Работа выполнена при поддержке гранта РФФИ № 18–07-00504_А.

Evaluation of the Relative Permeability Hysteresis Effect for the Thermo-Gas EOR Method with the Water-Air Mixture Injection

D.T. Mironov[1], P.V. Yalov[2]

Abstract. Issues of three-phase flow mathematical modeling for thermo-gas enhanced oil recovery method with water-air mixture injection - are discussed. 2D model simulation of thermo-gas method with water-air mixture injection in the kerogen content bazhenov suite rock types with average geological parameters and non-drainable zones, have shown the relative permeability hysteresis effect on the applied technology efficiency.

Keywords: thermal-gas treatment, EOR methods, bazhenov formation, oxygen-containing mixture injection, kerogen, miscible drive, wet in-situ combustion, enhanced oil recovery, relative permeability hysteresis.

Литература

1. В.Ю. Алекперов, В.И. Грайфер, Н.М. Николаев, В.Б. Карпов, В.И. Кокорев, Р.Г. Нургалиев, А.П. Палий, А.А. Боксерман, В.А. Клинчев, А.В. Фомкин. Новый отечественный способ разработки месторождений баженовской свиты (часть 1). «Нефтяное хозяйство» № 12 (2013), 100–105.

2. Д.Т. Миронов, П.В. Ялов. Оценка вовлечения в разработку недренируемых керогенсодержащих зон баженовской свиты при моделировании термогазового воздействии. «Труды НИИСИ», Т. 9 (2019), № 6, 94–83.

3. Д.Т. Миронов, А.А. Глушаков, П.В. Ялов. Влияние закачки водовоздушной смеси на эффективность термогазового воздействия для баженовской свиты при 2D моделировании. «Труды НИИСИ», Т. 9 (2019), № 6, 104–110.

4. J. E. Killough. Reservoir simulation with history-dependent saturation functions. «Society of Petroleum Engineers Journal» 16 (1976), 37-48.

5. SCHLUMBERGER, ECLIPSE User Manual. "Technical description" Schlumberger Ltd (2009), 519-538.

6. G. R. Jerauld. General three-phase relative permeability model for Prudhoe Bay. «SPE reservoir Engineering» T.12 (1997), 255-263.

7. Y. Ghomian, G. Pope, K. Sepehrnoori. Hysteresis and Field-Scale Optimization of WAG Injection for Coupled CO2-EOR and Sequestration. SPE 110639, Symposium on Improved Oil Recovery. 2008.

Разработка и тестирование многопроцессорной версии симулятора двухфазной фильтрации

М.Л. Сидоров[1], В.А. Пронин[1], В.Ю. Кузнецов[1], И.В. Афанаскин[2], А.В. Королев[2], П.В. Ялов[2]

[1]ФГУП «РФЯЦ–ВНИИЭФ», Саров, Россия, virtualvi@mail.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, ivan@afanaskin.ru

Аннотация. В статье приведены результаты тестирования распараллеленной модели двухфазной фильтрации слабосжимаемых жидкостей (нефти и воды) в упругом пласте с использованием неструктурированных сеток в модели распределенной памяти. Представлены модели двух реальных месторождений со сложной историей разработки и десятками скважин. Дано сравнение показателей разработки, рассчитанных в многопроцессорной версии, с результатами, полученными по модели в однопроцессорном режиме.

Ключевые слова: параллельные вычисления, неструктурированная сетка, модель распределенной памяти, интерфейс MPI, двухфазная фильтрация

1. Введение

Разработка месторождений углеводородов представляет собой комплексную проблему, для решения которой требуется использовать высокопроизводительные вычислительные системы или суперкомпьютеры. Актуальность использования суперкомпьютеров обусловлена разработкой гигантских и крупных нефтяных месторождений, а также многовариантностью проведения расчетов для обоснования оптимального режима разработки. Использование суперкомпьютеров позволяет проводить гидродинамическое моделирование всего месторождения в целом (с полным использованием геологической модели без огрубления сетки), а также увеличить точность расчетов с сокращением сроков их проведения.

Для решения задач на современных вычислительных системах требуется создавать специфические программы, которые будут учитывать особенности и ресурсы вычислительной машины. Такая программа разбивает вычислительный процесс на подзадачи, выполняемые одновременно и независимо с последующим обменом информации между вычислительными ядрами.

2. Разработка многопроцессорной версии симулятора

На базе методики описанной в [1] была разработана многопроцессорная версия симулятора на языке C++ по моделированию двухфазной фильтрации (НИМФА). Схема работы реализованного симулятора представлена на рис. 1.

Рис. 1. Схема многопроцессорной версии симулятора

Подробно принципы параллельного расчета двухфазной фильтрации слабосжимаемых жидкостей в упругом пласте приведены в [2]. Там же рассмотрены методы распараллеливания модели двухфазной фильтрации с использованием неструктурированных сеток в модели распределенной памяти, представлены основные принципы распараллеливания, а также методы распараллеливания явных и неявных уравнений. Особое внимание уделено распараллеливанию алгоритма расчета фильтрации с неявным учетом скважин. Приведены простые примеры. Настоящая работа посвящена разбору примеров моделирования реальных месторождений.

3. Тестовые задачи

В качестве тестовых задач использовались модели двух реальных месторождений.

Модель 1

На рис. 2 приведен общий вид первой модели нефтенасыщенного пласта. Месторождение принадлежит Северо-Кавказской нефтегазоносной провинции. Залежь нефти массивного типа. Коллектор терригенный. Нефть легкая, маловязкая с большим количеством растворенного газа (204 $м^3/м^3$). Абсолютная проницаемость пласта – 73 мД, пористость – 0,22 д.ед., расчлененность – 3,6, песчанистость – 0,62 д.ед., эффективная нефтенасыщенная толщина – 5,6 м, начальная нефтенасыщенность – 0,71 д.ед. Пластовая температура – 147 ^{0}C. Начальное пластовое давление – 339 атм при давлении насыщения – 18,8 атм, что дает большой запас упругой энергии. Система поддержания пластового давления отсутствует. Разработка ведется на упруго-водонапорном режиме.

Параметры модели:

- Период моделирования - 71 год.
- Область моделирования – 5,1 км * 3,7 км * 27 м.
- Количество скважин - 12 шт.
- Количество ячеек - 47709 (активных около 33 тысяч).
- История разработки - 40 лет (еще почти 40 лет - прогноз).
- Использовалась полностью неявная схема решения (с неявным учетом скважин).

PVT свойства для воды и нефти указаны в табл. 1 и 2 соответственно.

Плотности жидкостей при стандартных условиях равны: для воды - 1056 $кг/м^3$ и для нефти 815 - $кг/м^3$.

В табл. 3 представлены значения функций относительных фазовых проницаемостей (ОФП) для двухфазной фильтрации нефти и воды. Капиллярным давлением пренебрегается.

Таблица 1. PVT-свойства для воды, модель 1

Давление (атм)	B_w **($м^3/м^3$)**	**Сжимаемость (1/атм)**	**Вязкость (сПз)**
339	1,02	4,7E-05	0,36

Таблица 2. PVT-свойства для нефти, модель 1

Давление (атм)	B_o **($м^3/м^3$)**	**Сжимаемость (1/атм)**	**Вязкость (сПз)**
339	1,55	1,01E-04	0,397

Таблица 3. Функции относительных фазовых проницаемостей, модель 1

S_w, **д.ед.**	k_{rw}, **д.ед.**	k_{row}, **д.ед.**
0,37	0	1
0,4046	0,013	0,242
0,4253	0,023	0,131
0,459	0,036	0,074
0,525	0,078	0,029
0,557	0,103	0,012
0,5839	0,143	0,01
0,6	0,191	0,008
0,63	0,502	0

Сжимаемость породы составляет 4,7E-05 1/атм.

В качестве граничного условия для боковых сеточных блоков (рис. 3) задан водоносный пласт по модели Фетковича [3].

Исходная геологическая сетка была предоставлена в формате «Corner Point».

В расчете использовалась гексагональная сетка с регулярным шаблоном, хранящаяся в симуляторе в виде неструктурированной многогранной сетки. Решение матрицы СЛАУ проводилось посредством библиотеки LParSol [4], разработки ФГУП «РФЯЦ-ВНИИЭФ». Расчет был проведен на нескольких (1-8, рассмотрено 4 варианта) вычислительных ядрах с использованием параллельного интерфейса MPI.

В табл. 4 приведено сравнение ускорения расчета, полученного при использовании 2, 4 и 8 ядер.

Таблица 4. Время счета и ускорение при распараллеливании, модель 1

Число ядер	1	2	4	8
Время счета, с	602	346	289	1823
Ускорение	1	1,74	2,08	3,3

На рис. 4 и 5 приведена декомпозиция задачи на 4 и 8 параобластей.

На рис. 6 и 7 приведены изображения полей нефтенасыщенности, полученных в последовательном и параллельном режимах (на 8 вычислительных ядрах) на конец расчета.

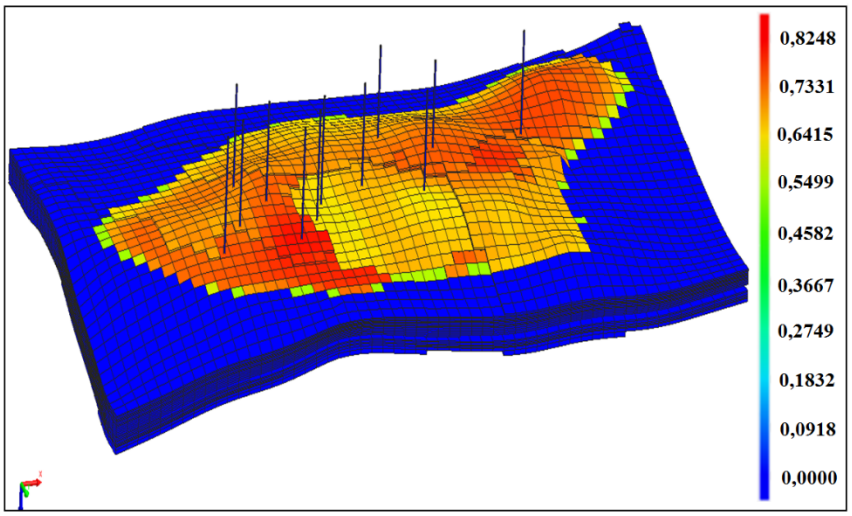


Рис. 2. Общий вид модели (поле нефтенасыщенности на начальный момент времени, д.ед.), модель 1

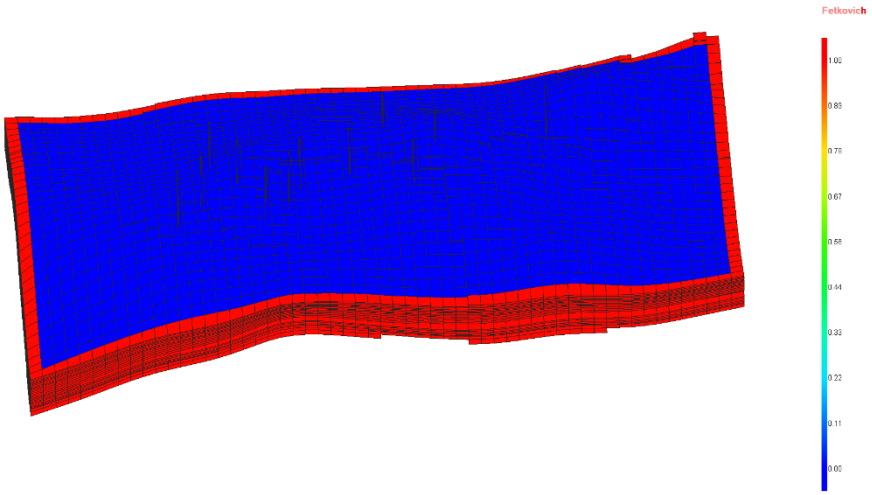


Рис. 3. Ячейки, граничащие с водоносным пластом, модель 1

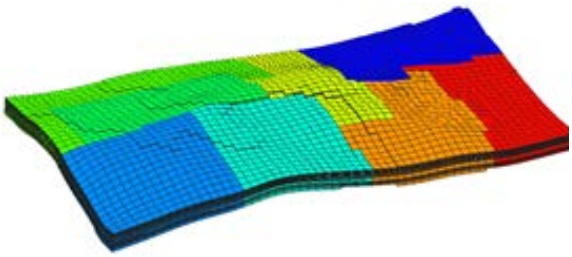


Рис. 4. Декомпозиция на 8 параобластей, модель 1

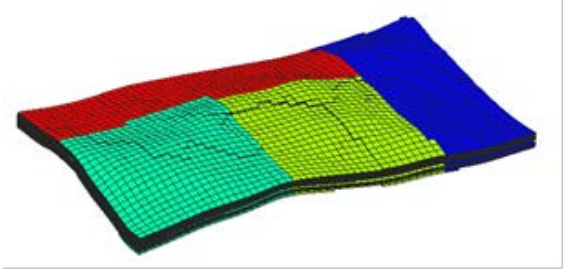


Рис. 5. Декомпозиция на 4 параобласти, модель 1

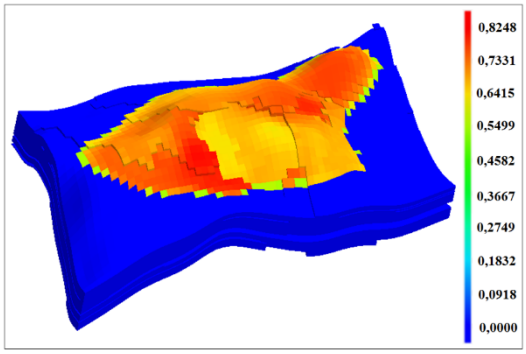


Рис. 6. Поле нефтенасыщенности (д.ед.), полученное в последовательном режиме, модель 1

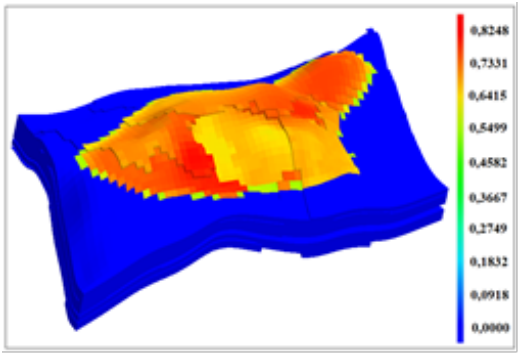


Рис. 7. Поле нефтенасыщенности (д.ед.), полученное в параллельном режиме (8 ядер), модель 1

Как видно из рис. 6-7 - поля нефтенасыщенности, полученные в результате последовательного и параллельного расчетов, не отличаются друг от друга (численные отличия в пределах машинной точности).

На рис. 8 и 9 приведены графики зависимости накопленной добычи нефти по всем скважинам и отдельно по одной из скважин соответственно, на рис. 10 представлен приток воды из водоносного горизонта. Результаты были получены в ходе параллельного расчета по реализованному симулятору и с помощью коммерческого программного обеспечения (КПО). Всплеск добычи после 40 лет объясняется проведением капитального ремонта скважин и реализацией мероприятий по интенсификации добычи нефти.

Из рис. 8-10 видно хорошее согласие результатов по реализованному симулятору и КПО

.

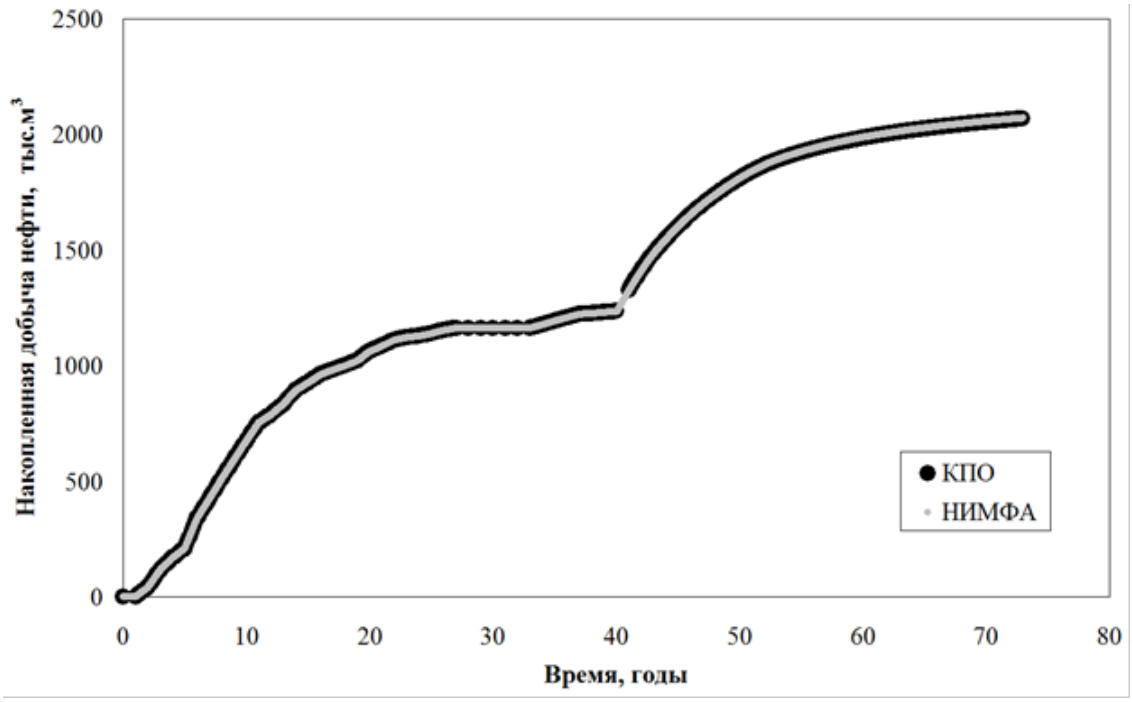


Рис. 8. Накопленная добыча нефти по всем скважинам, модель 1

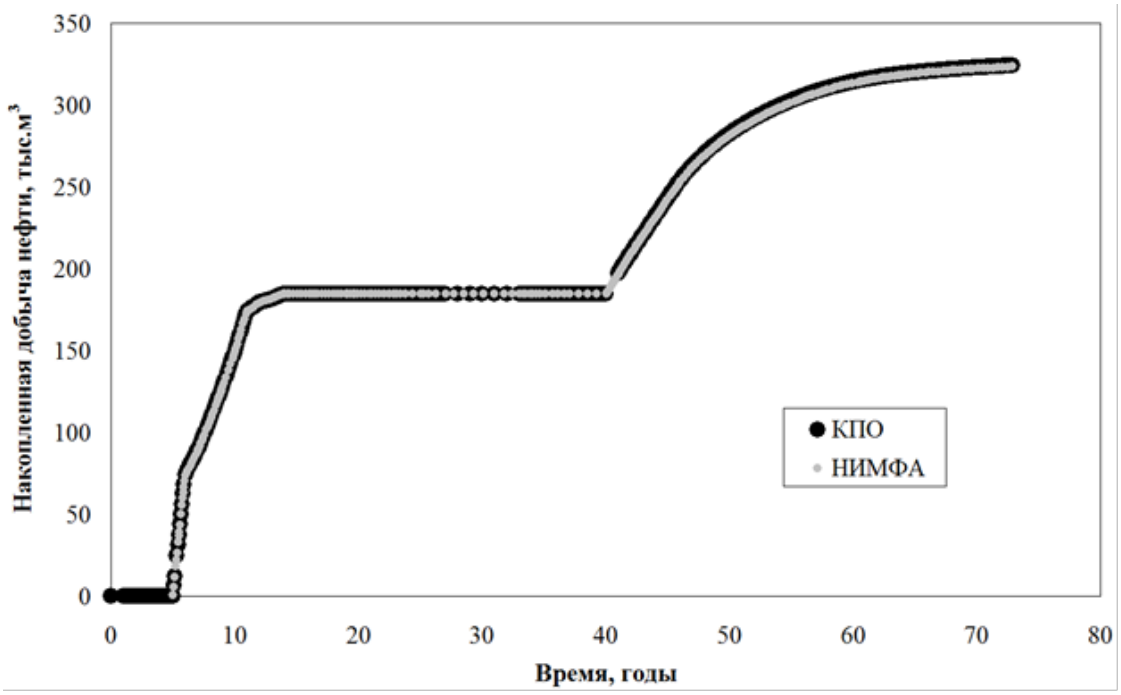


Рис. 9. Накопленная добыча нефти по одной из скважин, модель 1

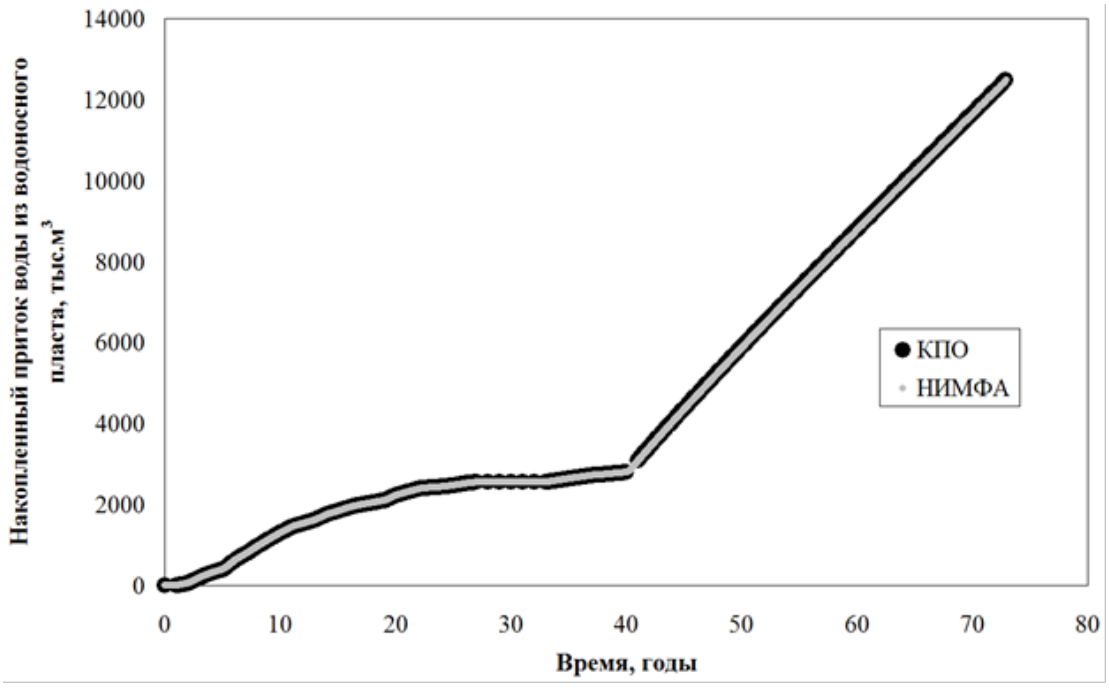


Рис. 10. Накопленный приток воды из законтурного водоносного пласта, модель 1

Модель 2

На рис. 11-12 приведен общий вид второй модели нефтенасыщенного пласта, представленного двумя разновеликими залежами. Месторождение принадлежит Западно-Сибирской нефтегазоносной провинции. Залежи нефти пластового типа с тектоническим нарушением, затухающим к центру основной залежи. Коллектор терригенный. Нефть легкая, маловязкая с умеренным количеством растворенного газа (50 $м^3/м^3$). Абсолютная проницаемость пласта – 103 мД, пористость – 0,12 д.ед., расчлененность – 5,3, песчанистость – 0,81 д.ед., начальная нефтенасыщенность – 0,65 д.ед. Начальное пластовое давление – 286 атм при давлении насыщения –

120 атм. В виду больших размеров основной залежи реализуется система поддержания пластового давления. Разработка ведется преимущественно на жестко-водонапорном режиме.

Параметры модели и режима разработки.

- Расчетный период в задаче 20 лет.
- Количество скважин - 165 шт. (из них 78 нагнетательных).
- Количество ячеек - 303 тыс. шт. (активных 265.5 тыс.)
- Использовалась полностью неявная схема решения (с неявным учетом скважин).

PVT-свойства воды и нефти приведены в табл. 5 и 6, ОФП и капиллярное давление - в табл. 7.

Таблица 5. PVT-свойства для воды, модель

Давление (атм)	B_w **($м^3/м^3$)**	**Сжимаемость (1/атм)**	**Вязкость (сПз)**
386	1,02	4,7E-05	0,33

Таблица 6. PVT-свойства для нефти, модель 2

Давление (атм)	B_o **($м^3/м^3$)**	**Сжимаемость (1/атм)**	**Вязкость (сПз)**
339	1,162	1,3E-04	0,62

Плотности жидкостей при стандартных условиях равны: для воды - 1011 кг/$м^3$ и для нефти 845 - кг/$м^3$.

Сжимаемость породы составляет 4,3E-05 1/атм.

Таблица 7. Функции относительных фазовых проницаемостей и капиллярного давления, модель 2

S_w**, д.ед.**	k_{rw}**, д.ед.**	k_{row}**, д.ед.**	P_{cow}**, атм**
0,34	0	0,6	2,9908
0,3503	0,0001	0,5551	0,4958
0,3809	0,0003	0,436	0,1743
0,4286	0,0024	0,2868	0,0863
0,4884	0,0102	0,1549	0,048
0,555	0,0287	0,0653	0,0274
0,6217	0,0612	0,0199	0,0152
0,6814	0,1048	0,0038	0,0078
0,7292	0,1512	0,0003	0,0032
0,7597	0,1868	0,0002	0,0008
0,7674	0,1967	0,0001	0,0002
0,77	0,2	0	0

Боковой внешний контур модели граничит с водоносным пластом, который описывается моделью Фетковича [3].

Модель была получена путем импорта данных в блочноцентрированном формате.

Особенностью данной модели является наличие затухающего разлома, рис. 12.

На рис. 13 представлена декомпозиция данной задачи на 4 параобласти.

В табл. 8 представлены времена счета на 1, 2, 4 и 8 вычислительных ядрах.

На рис. 14, 15 представлены поля водонасыщенности и давления на момент окончания расчета.

Таблица 8. Время счета и ускорение при распараллеливании, модель 2

Число ядер	1	2	4	8
Время счета, с	6520	4100	3030	2040
Ускорение	1	1,59	2,15	3,19

На рис. 16 приведен график зависимости накопленной добычи нефти по месторождению в целом. Результаты были получены в ходе параллельного расчета по реализованному симулятору и с помощью коммерческого программного обеспечения (КПО). Из приведенных графиков видно удовлетворительное согласие результатов по реализованному симулятору и КПО. Расхождение кривых на поздних временах происходит за счет краевых скважин и вызвано различным влиянием законтурной водоносной зоны в КПО и в НИМФА.

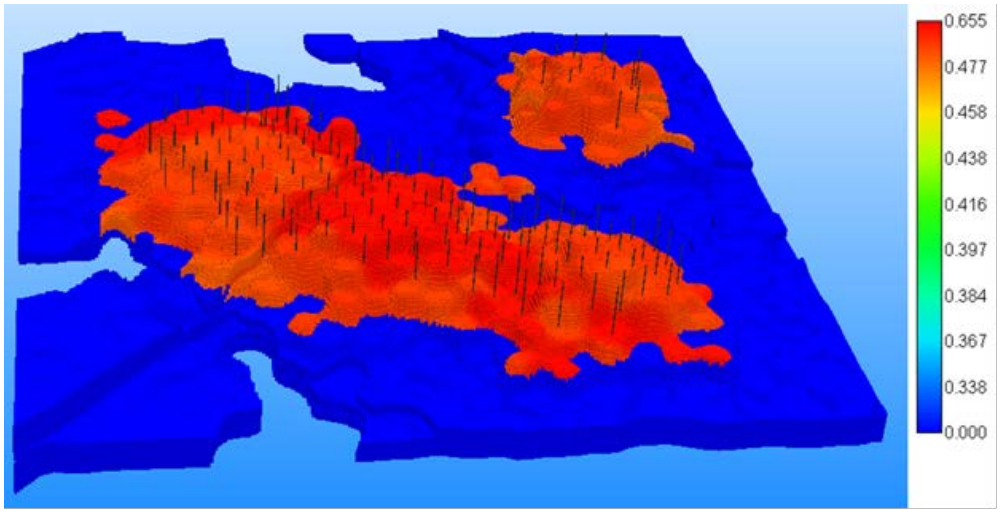


Рис. 11. Трехмерных вид на пласт (поле нефтенасыщенности на начальный момент времени, д.ед.), модель 2

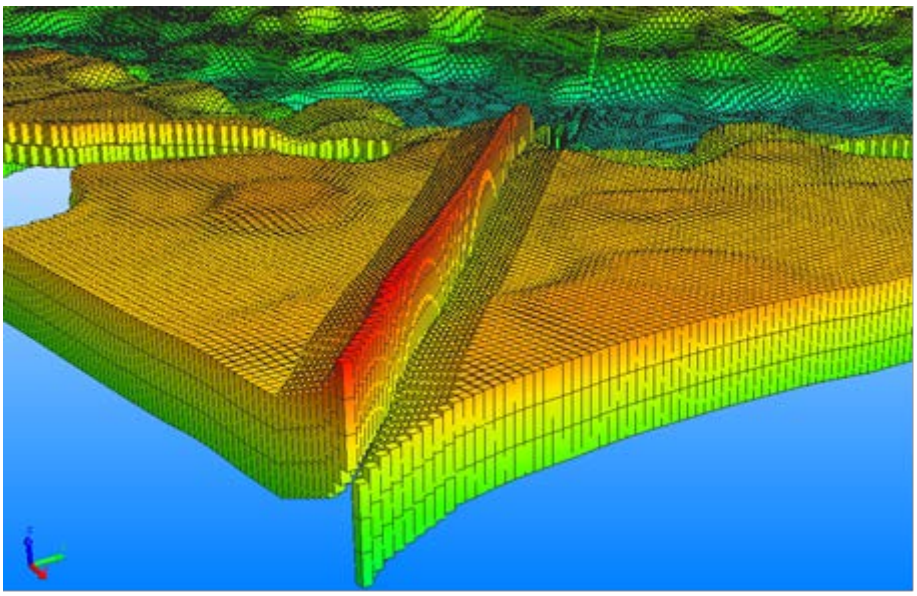


Рис. 12. Фрагмент модели в районе разлома, модель 2

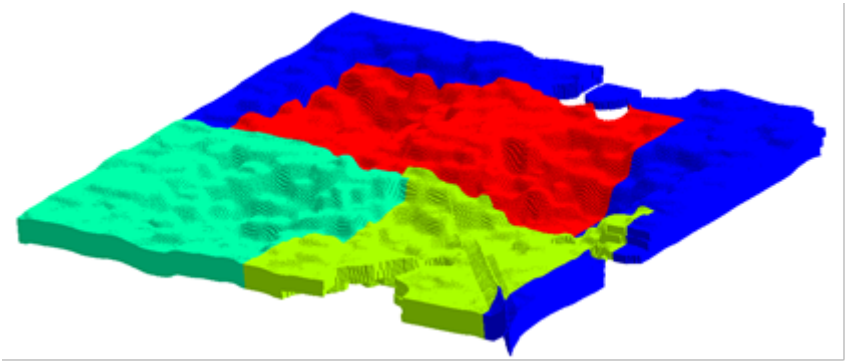


Рис. 13. Декомпозиция модели на 4 параобласти, модель 2

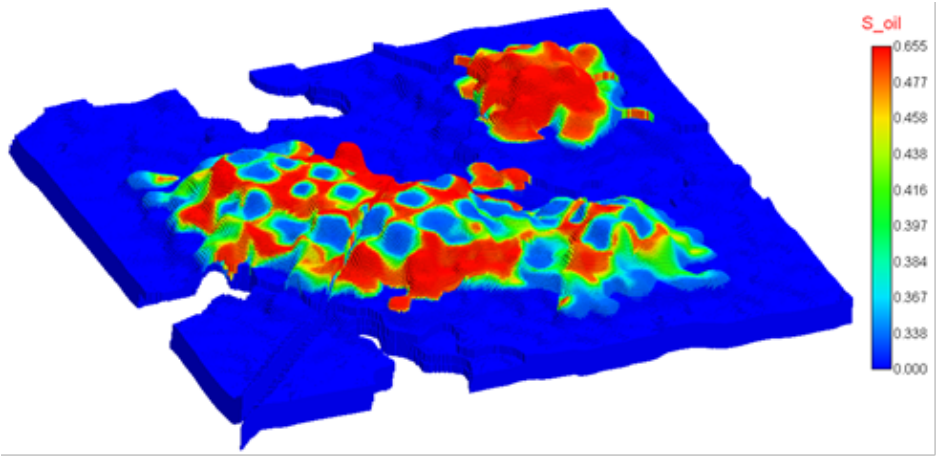


Рис. 14. Поле нефтенасыщенности на момент окончания расчета, д.ед., модель 2

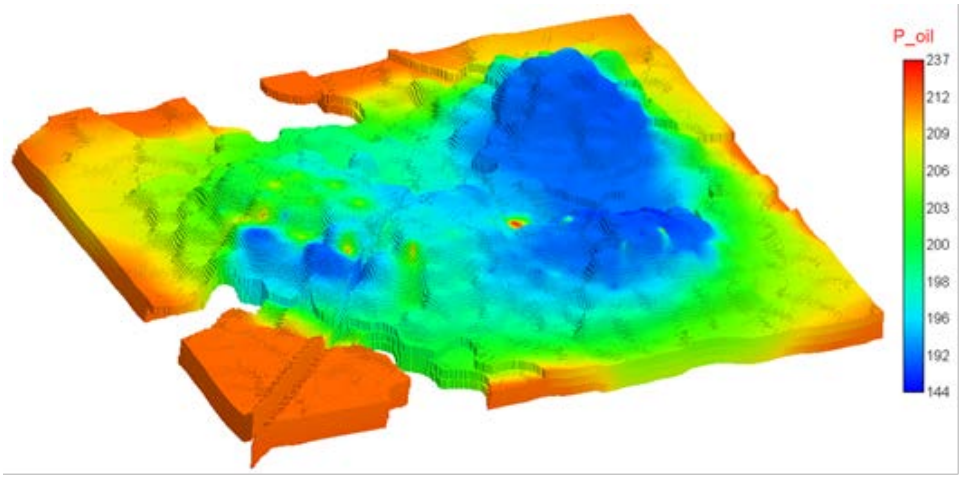


Рис. 15. Поле давления на момент окончания расчета, атм., модель 2

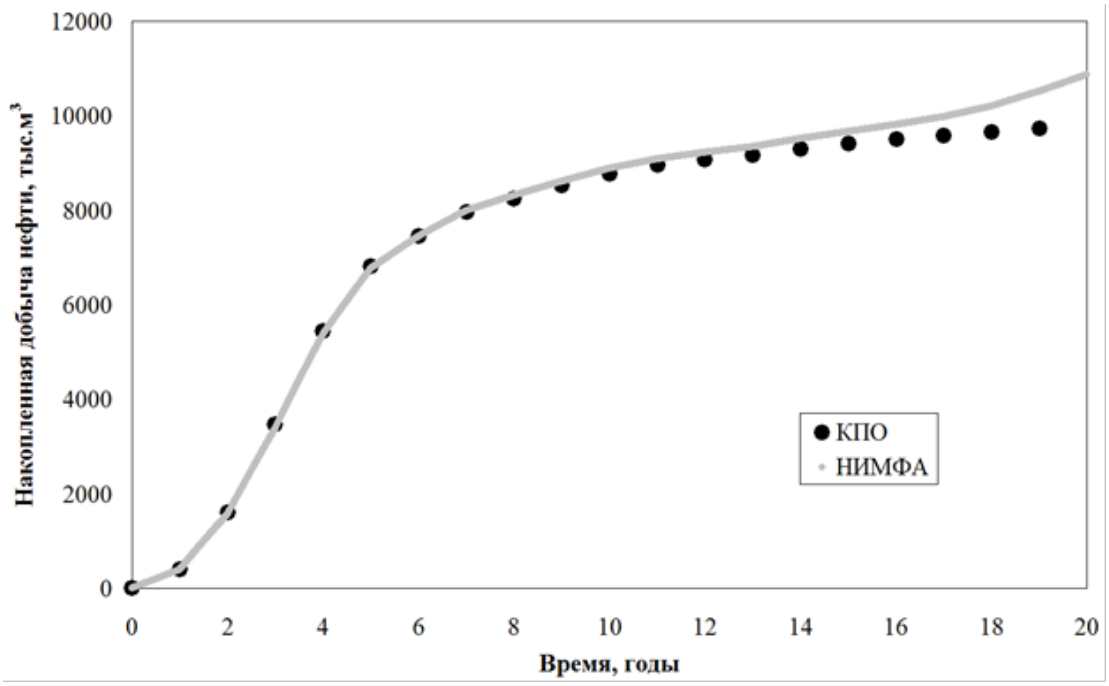


Рис. 16. Накопленная добыча нефти по всем скважинам, модель 2

4. Заключение

В статье изложены результаты тестирования многопроцессорной версии симулятора двухфазной фильтрации слабосжимаемых жидкостей в упругом пласте в модели распределенной памяти. Работоспособность распараллеленной методики подтверждена расчетами реальных задач в параллельном режиме.

Представлены основные результаты расчета двух реальных моделей и статистика по ускорению расчета. Результаты сравниваются с результатами расчета в однопроцессорном режиме по коммерческому ПО, получено качественное согласие. Результаты расчетов позволяют сделать следующие выводы:

- Реализованные методы решения уравнений двухфазной фильтрации и учета скважин в параллельном режиме дают результаты, которые согласуются с результатами, полученными по коммерческому программному обеспечению.
- Получено ускорение пропорциональное числу используемых в расчете ядер.

Работа выполнена при поддержке гранта РФФИ 18-07-00671 А.

Development and Testing of Two-Phase Filtration Multiprocessor Version Simulator

M.L. Sidorov, V.A. Pronin, V.Yu. Kuznetsov, I.V. Afanaskin, A.V. Korolev, P.V. Yalov

Abstract. Article describes test results of two-phase low-compressible fluids filtration model parallel calculation which involves non-structured grids of distributed memory model. Models of two real fields presented, which includes complicated production history and few dozens of wells. Production historie's comparison overviewed for multi- and monoprocessor calculations variants.

Keywords: parallel calculation, non-structured grid, distributed memory model, MPI interface), two-phase flow

Литература

1. Бутнев О.И., Кузнецов В.Ю., Пронин В.А., Сидоров М.Л., Кац Р.М., Афанаскин И.В., Королев А.В., Ялов П.В. Методика расчета двухфазной фильтрации жидкости в пористых средах на призматических сетках различного вида. «Вестник кибернетики», 2018, №3 (31), 135-152.

2. Сидоров М.Л., Пронин В.А., Кузнецов В.Ю., Афанаскин И.В., Королев А.В., Ялов П.В. Разработка параллельных методов расчета двухфазной фильтрации слабосжимаемых жидкостей в упругом пласте. «Нефтепромысловое дело», 2020, № 11 (623).

3. Aziz K, Odeh A.S. Comparison of Solution to a Three-Dimensional Black-Oil Reservoir Simulation Problem. JPT, 1981, Vol. 33, 13-25.

4. Артемьев А. Н., Бартенев Ю.Г., Басалов В.Г., Бондаренко Ю.А., Варгин А.М., Голубев А.А., Ерзунов В.А., Ломтев А.В., Максимов А.С., Панов А.И., Прокофьев А.И., Романова М.Д., Фролова Н.В., Щаникова Е.Б.. Библиотека решателей разреженных линейных систем. «Труды РФЯЦ-ВНИИЭФ», 2004, выпуск 7, 80-95.

Оптимизация операции перемножения матриц на основе технологии OpenCL

А.А. Бурцев[1]

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

Аннотация. В статье даётся первоначальное знакомство с технологией OpenCL, позволяющей использовать мощные ресурсы графических процессоров для повышения быстродействия вычислительных программ. Рассматриваются типичные приёмы разработки в среде OpenCL эффективных параллельных программ для обработки векторов и матриц. Основное внимание уделяется анализу различных способов построения программы для оптимизации операции перемножения больших матриц.

Ключевые слова: параллельное программирование, технология OpenCL, гетерогенные системы, микропроцессоры семейства КОМДИВ, операция перемножения матриц.

1. Введение

В современных высокопроизводительных системах для ускорения вычислений, как правило, применяется тот или иной вид параллельной обработки. При этом, конечно же, предполагается, что разработанная таким способом параллельная программа решения вычислительной задачи будет исполняться не одним, а сразу несколькими процессорами. Ожидается, что в результате совместной параллельной работы многих процессоров можно добиться значительного (в разы!) ускорения решения вычислительной задачи.

Понятно, что таким способом добиться значительного ускорения можно лишь для тех вычислительных задач, алгоритм решения которых поддаётся распараллеливанию, т.е. разбиению на такие участки, которые могут исполняться разными процессорами параллельно или одновременно (т.е. в одно и то же время).

Для распараллеливания вычислительных программ применяются самые разнообразные технологии [1,2]. Для разработки параллельной программы, призванной исполняться на разных процессорах (процессорных ядрах) одного компьютера с общей памятью применяется технология OpenMP [1,п.5.1; 2,гл.3]. Для построения распределённой программы, исполнение которой предполагается на семействе (кластере) компьютеров, связанных сетью, предложена технология MPI [1,п.5.2; 2,гл.4].

Для ускорения вычислений в рамках одного компьютера предлагаются также специализированные сопроцессоры SIMD класса. В микропроцессорном семействе КОМДИВ, разрабатываемом в ФНЦ НИИСИ РАН, к таким относятся векторный сопроцессор CPV и сопроцессор цифровой обработки сигналов CP2. Проведёнными исследованиями [3-5] было подтверждено, что такие математические спецпроцессоры позволяют в несколько раз повысить производительность определённого класса вычислительных задач, в частности, задач линейной алгебры [3] и цифровой обработки сигналов [4].

Относительно недавно обозначилось ещё одно направление исследований, обещающее значительное ускорение вычислительной программы за счёт её особого распараллеливания. Это направление связано с использованием в вычислительных программах ресурсов мощных графических процессоров. Такие специализированные процессоры для ускорения цифровой обработки изображений и видеоинформации стали встраивать в видеокарты (или графические адаптеры), управляющие формированием видеосигнала, поступающего на монитор компьютера.

Впоследствии ряд исследователей обратил внимание, что современные видеокарты содержат в своём составе огромное количество (сотни и даже тысячи) однородных специализированных процессоров, которые можно ухитриться использовать как массив параллельно функционирующих вычислительных ядер для ускорения обработки одной вычислительной программы. Первыми применить графические карты как ускорители вычислений предложили сотрудники компании NVIDIA, разработав для своих видеокарт технологию **CUDA** (**C**ompute **U**nified **D**evice **A**rchitecture). Затем аналогичную технологию, но уже пригодную для видеокарт различных семейств, предложила компания Apple. Впоследствии эту технологию приняла за основу и довела до стандарта Khronos Group (организация, занимающаяся выработкой открытых стандартов), дав ей название **OpenCL** (**Open C**omputing **L**anguage).

С тех пор OpenCL служит стандартом [6] разработки параллельных программ для так называемых гетерогенных (неоднородных) систем, в

которых наряду с обычными универсальными процессорами (CPU) для исполнения программы могут использоваться специализированные процессоры самого разнообразного характера, в том числе графические (GPU), а также процессоры обработки сигналов (DSP).

В новую микросхему семейства КОМДИВ [7], разрабатываемую в настоящее время в ФНЦ НИИСИ РАН, включёно графическое ядро с поддержкой технологии OpenCL (версии 1.2/1.1). А такое ядро можно использовать не только для ускоренной обработки видеоинформации, но и для ускорения вычислительных задач самого разнообразного характера.

Для испытания возможностей графического ядра этой микросхемы в НИИСИ на основе технологии OpenCL был разработан ряд прикладных программ обработки векторов и матриц, характерных для задач линейной алгебры. Полученные в результате прогонов этих программ показатели производительности свидетельствуют о том, что с помощью технологии OpenCL можно получить значительный выигрыш в ускорении решения подобных вычислительных задач.

В данной статье сначала предлагается первоначальное знакомство с технологией OpenCL. А затем рассматриваются характерные приёмы разработки эффективных программ по технологии OpenCL на примерах решения некоторых задач из области линейной алгебры. А также приводятся количественные показатели ускорения вычислительных операций над векторами и матрицами, которые в результате такой разработки удалось получить в среде OpenCL, поддерживаемой в настоящее время микросхемой графического ускорителя семейства КОМДИВ.

2. Первоначальное знакомство с технологией OpenCL

Кратко охарактеризуем, что первоначально надо знать про OpenCL программисту, чтобы быстро преобразовать свою вычислительную программу, подлежащую распараллеливанию, в полноценное OpenCL-приложение, способное существенно ускорить выполнение требуемой вычислительной задачи при его исполнении на устройствах, охваченных технологией OpenCL.

2.1. Основные понятия и особенности технологии OpenCL

Аппаратное окружение или OpenCL-среда, в которой будет исполняться OpenCL-программа, т.е. программа, разработанная по технологии OpenCL, состоит из совокупности нескольких OpenCL-устройств и одного головного устройства (см. рис. 1), так называемого хоста (host), роль которого может исполнять только универсальный процессор (CPU). Для простоты будем далее считать, что в системе помимо CPU есть видеокарта – устройство с графическим процессором (GPU), которое и будет исполнять роль OpenCL-устройства.

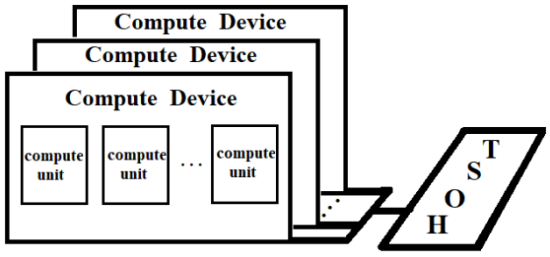


Рис. 1. Аппаратная конфигурация OpenCL

OpenCL-устройство, в свою очередь, содержит в своём составе массу однородных обрабатывающих элементов (processing elements), которые могут использоваться в качестве вычислительных ядер (OpenCL-программой). Можно представлять, что эти элементы организованы в одно-, двух- или даже трёхмерный массив. Каждый из этих элементов имеет свой процессор и внутреннюю (private) память, доступную только ему. Всем этим элементам доступна также общая глобальная (global) память, в которую и загружаются (с хоста) данные, подлежащие параллельной обработке, и из которой потом выгружаются обратно (в хост) результаты.

В зависимости от аппаратной реализации обрабатывающие элементы OpenCL-устройства могут быть сосредоточены в отдельных блоках – модулях (compute units), имеющих общую так называемую локальную (local) память (см. рис. 2). Такая память позволяет собранным в один модуль элементам хранить общие данные, а также обмениваться информацией между собой.

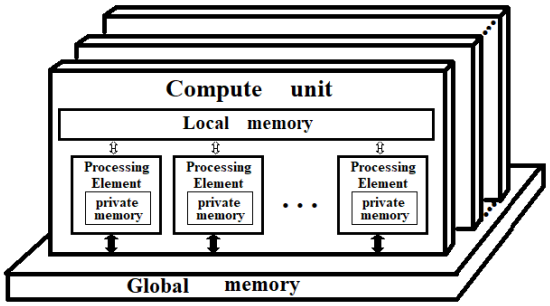


Рис. 2. Строение OpenCL-устройства

Таким образом, каждый обрабатывающий элемент может обращаться (читать и записывать) к памяти трёх видов. Самый быстрый доступ ему обеспечивается к своей внутренней памяти небольшого размера. Локальная память модуля значительно больше по объёму, но доступ к ней осуществляется для всех элементов модуля

и оказывается более медленным. Но самой медлительной по скорости доступа и наибольшей по размеру выступает глобальная память всего устройства. Вот такую иерархию памяти приходится учитывать при программировании действий, выполняемых на обрабатывающих элементах.

Параллельная программа, разработанная по технологии OpenCL, (или OpenCL-программа) состоит как минимум из двух программных компонент: OpenCL-приложения, запускаемого на хосте, и особо оформленной одной или нескольких процедур, каждая из которых будет запускаться параллельно сразу на всех обрабатывающих элементах OpenCL-устройства. Такие особые процедуры, называемые также OpenCL-ядрами (kernels), составляются на специально разработанном языке, представляющем собой некий диалект языка Си (именно этот язык и был первоначально назван OpenCL).

В терминах OpenCL-программы параллельные процессы (нити или треды), каждый их которых выполняет процедуру OpenCL-ядра, называются рабочими элементами (work-item). Для функционирования каждого рабочего элемента OpenCL-программы в идеале должен быть назначен отдельный обрабатывающий элемент аппаратуры OpenCL-устройства. Но количество рабочих элементов в программе может оказаться больше имеющегося числа обрабатывающих элементов в аппаратуре. Тогда OpenCL-программа будет развиваться в режиме попеременного использования обрабатывающих элементов для исполнения ими действий, предусмотренных теми или иными рабочими элементами.

Задаваемую аппаратной конфигурацией возможность размещения вычислительных ядер в отдельных модулях с общей для них локальной памятью можно учитывать при построении параллельной OpenCL-программы, объединяя рабочие элементы, которым требуется общая локальная память, в так называемые рабочие группы (WorkGroup). Внутри каждой рабочей группы элементы также можно рассматривать организованными в массив (одно-, двух- или трёхмерный).

При исполнения OpenCL-ядра каждый рабочий элемент может узнать, под каким номером (парой или тройкой номеров) он индексируется в глобальном пространстве всех рабочих элементов, организованных в массив, а также узнать, как он индексируется в локальном пространстве своей рабочей группы. Для этого в библиотеке языка OpenCL предусмотрены соответствующие Си-функции. Используя такие функции и получив номера-индексы для идентификации своего места в локальной группе, а также в общем пуле всех рабочих элементов, OpenCL-процедура ядра может задать особое поведение своего рабочего элемента.

Далее на примере построения простой программы параллельной обработки векторов вещественных чисел рассмотрим, как оформляется процедура OpenCL-ядра и какие самые необходимые действия должны быть обязательно исполнены OpenCL-приложением, запускаемым на хост-процессоре.

2.2. Пример процедуры OpenCL-ядра

В качестве примера возьмёмся составить OpenCL-программу для выполнения операции сложения двух векторов **AXPY**, применяемой часто при решении задач линейной алгебры. Такую операцию (в обычной последовательной программе) можно выразить Си-функцией:

```
#define Alfa 2.7 //зададим Alfa как константу
void sAXPY(int N,float *X,float *Y){
 int k;
 for (k = 0; k < N; k++)
   Y[k] = Alfa * X[k] + Y[k]; /*/
}//sAXPY
```

Теперь предположим, что в параллельной OpenCL-программе можно будет задействовать ровно N рабочих элементов. Тогда можно легко составить процедуру OpenCL-ядра (на языке OpenCL), поручив k-му рабочему элементу исполнять ключевое действие (помеченное **/*/**) для k-х элементов заданных векторов (X,Y):

```
#define Alfa (2.7)
__kernel void axpy_ (
  __global const float *X,
  __global float *Y ) {
  int k = get_global_id(0);
  Y[k] = Alfa * X[k] + Y[k]; /*/
}//axpy_
```

Поясним особые описания, выделенные в отдельных местах этой процедуры. Прежде всего, заметим, что заголовок процедуры ядра начинается с ключевого слова **__kernel**, а каждый её параметр предваряется ключевым словом **__global**, характеризующим вид памяти (глобальный), где располагаются подлежащие обработке вектора-параметры. Все локальные переменные процедуры ядра тоже могут быть объявлены с ключевым словом для указания вида памяти. При отсутствии такого указания считается, что такие переменные будут располагаться во внутренней (private) памяти.

Ключевое слово **const**, характеризующее параметр (X), применяется для указания, что доступ к такому объекту в глобальную память предполагается только по чтению. Подобное указание позволяет разместить данный объект в особой константной зоне глобальной памяти, обеспечив к нему более быстрый доступ. (Например, путём длительного удержания его в

кэш-памяти). А функция **get_global_id** позволяет узнать номер (k), под которым индексируется в массиве всех рабочих элементов тот, который исполняет в данный момент процедуру ядра.

Но чтобы все обрабатывающие элементы OpenCL-устройства могли совместно выполнить параллельную программу даже с такой простой OpenCL-процедурой ядра, требуется, чтобы OpenCL-приложение, запущенное на хосте, исполнило достаточно длинную цепочку типовых обязательных действий, для каждого из которых предусмотрена соответствующая функция OpenCL-библиотеки. Знакомство с этими функциями проведём в несколько этапов (см. п. 2.3-2.7).

2.3. Инициализация OpenCL-среды

Прежде всего, необходимо получить информацию о платформах OpenCL, которые поддерживаются на имеющейся компьютерной установке (а их может быть несколько), затем выбрать одну из них (выберем первую) и получить её дескриптор. Для этого надо воспользоваться функцией **clGetPlatformIDs**:

```
// узнаем число поддерживаемых платформ nPltf
cl_int err; cl_uint nPltf;
err=clGetPlatformIDs(0,NULL,&nPltf);
if(err!=CL_SUCCESS||nPltf<=0)
{ printf("No OpenCL platforms!\n");...
}
// получим дескриптор 1-ой по порядку платформы
cl_platform_id PltfId;
err=clGetPlatformIDs(1,&PltfId,NULL);
```

Почти каждая функция OpenCL-библиотеки возвращает код завершения (`err`). Прежде, чем продолжать дальнейшую работу, следует, вообще говоря, убедиться, что вызванная функция сработала нормально и возвращённый ею код равен значению `CL_SUCCESS`. Будем предполагать, что аналогичные проверки кода завершения выполняются далее повсеместно, но приводить их более не будем.

Получив дескриптор платформы (`PltfId`), теперь с помощью многократных вызовов функции **clGetPlatformInfo** можно узнать разнообразную информацию о ней: её название, производитель, какую поддерживает версию OpenCL и какие его расширения.

Платформа может охватывать несколько устройств различных типов. Требуется получить дескрипторы тех устройств, которые будут использованы для исполнения параллельной OpenCL-программы, и подготовить далее с их участием так называемый OpenCL-контекст, который будет целиком характеризовать среду исполнения процедурных ядер, т.е. основной части OpenCL-программы на этих устройствах. Для этого можно воспользоваться функциями **clGetDeviceIDs** и **clCreateContext**.

Но если использовать однотипные OpenCL-устройства (или вообще только одно), то можно предложить более простой способ образования OpenCL-контекста, основанный на применении функции **clCreateContextFromType** :

```
cl_context cntxt;
cl_context_properties cntxtPrpt[]=
 { CL_CONTEXT_PLATFORM, PltfId,0 };
cntxt= clCreateContextFromType
 (cntxtPrpt,CL_DEVICE_TYPE_GPU,
   NULL, NULL, &err); // err= код заверше-
ния
```

Здесь вызов этой функции создаст дескриптор контекста (`cntxt`), включив в него все подключённые к платформе графические процессоры (устройства типа `CL_DEVICE_TYPE_GPU`). В качестве 1-го параметра этого вызова указывается массив (`cntxtPrpt`), характеризующий свойства создаваемого контекста. Здесь такой массив задан в самом упрощённом виде: в нём указан лишь дескриптор платформы.

У созданного же контекста можно с помощью функции **clGetContextInfo** запросить информацию об используемых им устройствах и получить их дескрипторы.

```
cl_int nDv; // кол-во дескрипторов устройств
size_t DvBfsz; //размер всех дескрипторов
size_t szDv= sizeof(cl_device_id);
cl_device_id *Dev;
err= clGetContextInfo(cntxt, CL_CON-
TEXT_DEVICES,0,NULL,&DvBfsz);
nDev= DvBfsz/szDv;
DevId=(cl_device_id*)mal-
loc(nDv*szDv);
err= clGetContextInfo(cntxt, CL_CON-
TEXT_DEVICES,DvBfsz,DevId,NULL);
```

С помощью функции **clGetDeviceInfo** из дескриптора устройства можно получить много нужной информации о нём. Например, сколько в нём модулей (computer units):

```
// узнаем, сколько computer units у k-го устройства
cl_uint nCUnits; size_t size;
err= clGetDeviceInfo(DevId[k],
  CL_DEVICE_MAX_COMPUTE_UNITS,
  sizeof(cl_uint),&nCUnits,&size);
printf("%d compute units:\n",nCU-
nits);
```

Но основная потребность в дескрипторе устройства реализуется при создании очередей исполнения команд. В OpenCL-контексте, подготавливаемом для запуска процедурных ядер на OpenCL-устройствах, такая очередь должна быть создана для каждого используемого устройства. Ведь именно в неё и будут поступать команды устройству в процессе исполнения OpenCL-программы.

Для создания очереди используется функция

clCreateCommandQueue. Продемонстрируем, как можно построить очередь команд для устройства `DevId[0]` в контексте `cntxt` и получить её дескриптор (`cmndQueue`):

```
cl_command_queue cmndQueue;// дескриптор
cmndQueue= clCreateCommandQueue
 (cntxt, DevId[0], 0 ,&err);
```

С созданием очередей команд завершается тот начальный этап подготовки OpenCL-среды, который требуется осуществить перед исполнением основной части (запуском ядер) всякой OpenCL-программы.

Заметим, что выполняемые на этом этапе действия могут для разных OpenCL-программ оставаться одинаковыми, так как они почти не зависят от алгоритмической сущности параллельной программы, предназначенной для исполнения в среде OpenCL. А вот действия, которые предстоит далее выполнить на последующих этапах, для разных OpenCL-программ уже могут существенно различаться.

2.4. Компоновка OpenCL-ядер

Одна или несколько процедура ядра, объединённых вместе в одну программу на языке OpenCL, должны быть предварительно откомпилированы в некоторый универсальный внутренний код, который далее будет приниматься на исполнение драйвером OpenCL-устройства. Такая компиляции может быть осуществлена последовательными вызовами двух функций. Вызовом функции **clCreateProgramWihSource**:

```
cl_program prgrm
prgrm= clCreateProgramWithSource
(cntxt, 1, (const char**)&cSrcString,
  NULL, NULL);
```

в OpenCL-контекст подключается новый программный объект **prgrm**. В этом вызове 3-й параметр задаёт ссылку на массив указателей строк **cSrcString** исходного текста программы, а 2-й параметр задаёт количество строк. Можно собрать всю программу, подлежащую компиляции, в одну строку. Содержимое этой строки можно сформировать чтением из текстового файла (обычно с расширением *.cl) или просто задать как строку-константу, например:

```
char* cSrcString =
"#define Alfa (2.8) \n"\
"__kernel void axpy_ ( \n"\
" __global const float *X, \n"\
" __global float *Y ) { \n"\
" int k = get_global_id(0); \n"\
" Y[k]=Alfa*X[k]+Y[k]; /*/ \n"\
}//axpy_ \n"\
;
```

Далее полученный объект **prgrm** компилируется вызовом функции **clBuildProgram**:

```
err = clBuildProgram (prgrm,
      0, NULL, NULL, NULL, NULL);
```

Если при компиляции будут обнаружены ошибки программы, то код завершения (`err`) будет отличен от значения CL_SUCCESS. Тогда для уточнения причины ошибки можно вызвать функцию **clGetProgramBuildInfo**.

Откомпилированная программа может содержать несколько процедур ядер. Каждому ядру нужно сопоставить свой дескриптор, вызвав функцию **clCreateKernel** и задав в ней имя процедуры (2-м параметром):

```
cl_kernel krnl;
krnl=clCreateKer-
nel(prgrm,"axpy_",0);
```

Этот дескриптор (krnl) и будет далее использоваться при запуске процедуры ядра.

2.5. Подготовка OpenCL-данных

Если процедура ядра имеет параметры-массивы, то перед её запуском необходимо подготовить объекты, которые будут представлять выделенные в памяти OpenCL-устройства буфера, куда будут загружаться (до запуска процедуры ядра) значения элементов массивов, представляющие входные данные, и из которых впоследствии будут выгружаться значения, представляющие выходные данные. Для этого используется функция **clCreateBuffer**:

```
 cl_mem memObjX; cl_mem memObjY;
memObjX = clCreateBuffer
  (cntxt, CL_MEM_READ_ONLY,
   sizeof(float)*N,NULL,NULL);
memObjY = clCreateBuffer
  (cntxt, CL_MEM_READ_WRITE,
   sizeof(float)*N,NULL,NULL);
```

В ней 2-ым параметром задаётся, будут ли данные этого объекта доступны только по чтению, только по записи или по чтению и записи. В приведённом примере объект `memObjX` создаётся только для чтения, а объект `memObjY` – для чтения-записи.

Загрузка данных из основной памяти хоста в буфера объектов, созданные в глобальной памяти OpenCL-устройства, осуществляется вызовом функции **clEnqueueWriteBuffer**:

```
err= clEnqueueWriteBuffer
(cmndQueue, memObjX, CL_TRUE, 0,
 sizeof(float)*N, A, 0, NULL, NULL);
err = clEnqueueWriteBuffer
(cmndQueue, memObjY, CL_TRUE, 0,
 sizeof(float)*N, B, 0, NULL, NULL);
```

В этом примере устройству дано указание загрузить из основной памяти хоста вектор A[N] из N вещественных чисел (типа float) в буфер объекта `memObjX`, а вектор B[N] – в буфер объекта `memObjY`.

А чтобы выгрузить данные из буфера объекта

memObjY обратно в память хоста по адресу расположения вектора B[N], следует вызывать функцию **clEnqueueReadBuffer** (но уже после запуска процедуры ядра):

```
errNum = clEnqueueReadBuffer
 (cmmnQueue, memObjY, CL_TRUE, 0,
 sizeof(float)*N, B, 0, NULL, NULL);
```

2.6. Запуск процедуры OpenCL-ядра

Для выполнения операции AXPY над векторами A, B (B= Alfa*A+B) осталось лишь вызвать саму процедуру ядра **axpy_**, которая в OpenCL-программе представлена дескриптором krnl, назначив ей объекты memObjX и memObjY в качестве 1-го и 2-го параметра.

Для назначения каждого параметра процедуры ядра вызывается функция **clSetKernelArg**:

```
err= clSetKernelArg(krnl, 0,
        sizeof(cl_mem), &memObjX);
err= clSetKernelArg(krnl, 1,
        sizeof(cl_mem), &memObjY);
```

В ней 2-м аргументом задаётся номер параметра процедуры ядра (нумерация начинается с 0), для которого назначается в качестве фактического параметра объект, ссылка на который задаётся 4-м аргументом, а его размер – 3-им.

Для запуска ядра в работу вызывается функция **clEnqueueNDRangeKernel**:

```
size_t globWS[1] = { N };
err=clEnqueueNDRangeKernel(cmndQueue,

krnl,1,NULL,globWS,NULL,0,NULL,NULL);
```

В этом примере она запускает заданную дескриптором krnl процедуру ядра (**axpy_**) на всех N рабочих элементах, рассматривая их как одномерный массив: 3-й параметр такого вызова задаёт размерность 1, а 5-й параметр (globWS) указывает на массив размеров, где по 1-му измерению поставлено значение N.

На самом деле эта функция только инициирует такой запуск, т.е. ставит в очередь команд (cmndQueue) OpenCL-устройства действия, которые должны привести к запуску ядер. В то время, пока ядра будут исполняться на устройстве, OpenCL-программа хоста может осуществлять какие-то другие действия (например, запускать ядра на другом OpenCL-устройстве).

Но впоследствии OpenCL-программе хоста всё же придётся дождаться момента, когда на всех её рабочих элементах запущенная процедура ядра отработала. Осуществить такое ожидание можно разными способами. Проще всего вызвать функцию **clFinish** и убедиться, что в очереди команд этого устройства все команды уже отработали, т.е. завершены:

```
clFinish(cmndQueue);
```

2.7. Функции освобождения ресурсов

Для подготовки среды исполнения ядер OpenCL-программа хоста вызывает многие функции OpenCL-библиотеки для выделения необходимых ресурсов и создания разнообразных дескрипторов: контекстов, очередей команд, объектов программ, ядер, объектов памяти. При завершении OpenCL-программы все эти ресурсы должны быть освобождены. Для этого тоже предусмотрены соответствующие функции OpenCL-библиотеки:

```
clReleaseMemObject(memObjX);
clReleaseMemObject(memObjY);
clReleaseCommandQueue(cmndQueue);
clReleaseKernel(krnl);
clReleaseProgram(prgrm);
clReleaseContext(cntxt);
```

2.8. Измерение производительности

Чтобы узнать, какое ускорение вычислений обеспечивает применение технологии OpenCL, нужно уметь измерять время исполнения ядер OpenCL-процедур на OpenCL-устройствах и иметь возможность сравнивать его с тем, которое затрачивается на исполнение той же функции в обычной программе на одном процессоре.

Если требуется вести хронометраж времени при исполнении OpenCL-ядер, надо, во-первых, при вызове функции создании очереди команд задать особое значение для 2-го параметра:

```
cmndQueue=clCreateCommandQueue(cntxt,
DevId[0],CL_QUEUE_PROFILING_ENA-
BLE,0);
```

Во-вторых, при запуске процедуры ядра указать (последним параметром) событие, с которым будет связано её исполнение:

```
cl_event Ev;
err=clEnqueueNDRangeKernel(cmndQueue,
krnl,1,NULL,globWS,NULL,0,NULL,&Ev);
```

Впоследствии (уже после завершения всех действий на OpenCL-устройстве) это позволит узнать время старта и завершения процедуры ядра (на всех рабочих элементах), а значит, и посчитать время её исполнения. Такую информацию (с точностью до наносекунд) можно получить с помощью нескольких вызовов функции **clGetEventProfilingInfo**:

```
cl_ulong TimeStart, TimeFin;
err=clGetEventProfilingInfo(Ev,
 CL_PROFILING_COMMAND_START,
 sizeof(cl_ulong),&TimeStart,NULL);
err=clGetEventProfilingInfo(Ev,
 CL_PROFILING_COMMAND_END,
 sizeof(cl_ulong),&TimeFin,NULL);
cl_ulong RunTime= TimeFin-TimeStart;
```

Заметим, что полученное значение времени (RunTime) характеризует лишь время непосредственного исполнения ($T_Я$) процедуры ядра на OpenCL-устройстве.

Для подсчёта же полного времени (T_{CL}) ис-

полнения в подготовленной OpenCL-среде операции AXPY, аналогичной вызову функции sAXPY(A,B) на CPU, необходимо добавить к нему время загрузки векторов (A,B) в память устройства ($T_З$), назначения параметров процедуры ядра ($T_П$), её запуск с постановкой в очередь команд ($T_О$) и выгрузку вектора (B) обратно из памяти устройства ($T_В$):

$T_{CL} = T_З + T_П + T_О + T_Я + T_В$

Поэтому в дальнейшем будем сравнивать время исполнения операции на одном CPU (T_{CPU}) не только с чистым временем исполнения ($T_Я$) OpenCL-ядра (или ядер), но и с полным временем (T_{CL}), которое затрачивается на выполнение аналогичной операции в проинициализированной (согласно п.2.3) OpenCL-среде.

2.9. Результаты ускорения операций

Сначала попробуем оценить, насколько удалось с помощью технологии OpenCL ускорить операцию AXPY, взятую в качестве простого примера для составления OpenCL-программы.

Для получения количественных показателей полученного ускорения была разработана (первоначально в среде MS Visual Studio 2017 под ОС Windows-10) Си-программа, которая исполняла операцию AXPY(A,B) для векторов сначала на одном универсальном процессоре (CPU Intel-i59400 с частотой 2.9 Ггц), а затем в подготовленной среде OpenCL на графическом процессоре (GPU UHD 630 на частоте 350 МГц). Программа многократно прогонялась для больших векторов различной длины $N=10^k$ (k=3,4,5,6,7). Результаты этих прогонов представлены в таблице 1.

Таблица 1. Оценка ускорения операции AXPY

N	T_{CPU}	T_{CL}	$T_Я$	K1	K2
10^3	0.0021	2.5863	0.00708	0.001	0. 296
10^4	0.0175	2.4166	0.00633	0.007	2.763
10^5	0.1758	2.6274	0.03042	0.066	5.779
10^6	1.6980	4.6700	0.37358	0.364	4.545
10^7	17.4727	25.045	4.82750	0.698	3.619

T_{CPU} – время исполнения на CPU
T_{CL} – полное время исполнения в OpenCL
$T_Я$ – время исполнения OpenCL-ядра
(все времена даны в миллисекундах)
K1 – коэффициент ускорения = **T_{CPU} / T_{CL}**
K2 – коэффициент ускорения = **T_{CPU} / $T_Я$**

В ней коэффициент K2 показывает, что уже для векторов длиной 10^4 исполнение операции OpenCL-ядром ускоряет её в 2,7 раза, а на длине 10^5 – почти в 6 раз (5,779). Но из-за непроизводительных затрат полное время операции в среде OpenCL (на этой конфигурации аппаратуры) не удаётся ускорить: коэффициент K1 даже для векторов длиной 10^7 не достигает 1 (0,698).

Операция APXY хоть и хорошо поддаётся распараллеливанию, но не несёт в себе большой вычислительной нагрузки, ведь для каждой пары элементов (X_k,Y_k) исполняется лишь 2 арифметические команды (×,+). И получается, что львиная доля времени при исполнении этой операции уходит на передвижение по памяти массивов данных, подвергаемых обработке.

Попробуем усилить вычислительную нагрузку на обрабатываемые элементы и реализуем по технологии OpenCL такую операцию над двумя векторами, где на каждую пару элементов выполняется гораздо больше вычислительных действий. Пусть заголовок Си-функции такой операции остаётся прежним (с двумя векторами в качестве параметров), а вычисления над парой элементов этих векторов станут гораздо сложнее (см. оператор, помеченный **/*/**):

```
void SINCOS(int N,float *X,float *Y){
 int k;
 for (k = 0; k < N; k++)
   Y[k] = sin(X[k]) + cos(Y[k]); /*/
}//SINCOS
```

Для исполнения такой операции в OpenCL-среде составим процедуру ядра:

```
__kernel void sincos_ (
  __global const float *X,
  __global float *Y ) {
  int k = get_global_id(0);
  Y[k]= sin(X[k]) + cos(Y[k]); /*/
}//sincos_
```

И поместим её в строку **cSrcString**, как это сделали раньше для процедуры axpy_ (в п. 2.4). В уже составленном OpenCL-приложении для хоста изменим только фрагмент с вызовом функции **clCreateKernel**:

```
krnl=clCreateKer-
nel(prgrm,"sincos_",0;
```

И теперь OpenCL-программа готова к исполнению уже с новой процедурой ядра.

Прогоним её несколько раз для векторов различной длины $N=10^k$ (k=3,4,5,6,7) и проанализируем полученные результаты, представленные в таблице 2. Судя по её коэффициенту K2, исполнение вычислительной операции OpenCL-ядром уже для векторов длиной 10^3 ускоряет её в 7 раз, а на длине 10^7 – почти в 30 раз. Полное же время этой операции в данной OpenCL-среде удаётся ускорить уже для векторов длиной 10^5 (1.92), а для векторов длиной 10^7 можно обеспечить такое ускорение почти в 17 раз.

Таблица 2. Оценка ускорения операции SINCOS

N	T_{CPU}	T_{CL}	$T_Я$	K1	K2
10^3	0.0570	2.7991	0.00081	0.020	7.0518
10^4	0.4973	2.2940	0.02108	0.217	23.588
10^5	5.6228	2.9217	0.24317	1.924	23.123
10^6	71.897	6.3565	2.41117	11.31	29.819
10^7	731.533	43.829	23.6200	16.69	30.971

Полученное первоначальное знакомство с

OpenCL позволяет сделать вывод, что с помощью этой технологии можно значительно ускорить те вычислительные задачи, которые относительно легко можно подвергнуть массовому распараллеливанию, и в которых число команд вычислительного характера значительно превосходит количество команд передвижения данных по памяти.

3. Ускорение перемножения матриц с помощью OpenCL

В задачах линейной алгебры часто используется операция перемножения матриц. Она характеризуется вычислительной сложностью $O(N^3)$ и сложностью по работе с памятью $O(N^2)$. Это значит, что число вычислительных команд, которое потребуется исполнить для перемножения матриц размера N×N, пропорционально величине N^3, а число команд обращений к памяти пропорционально величине N^2. Поэтому можно надеяться, что с помощью технологии OpenCL удастся значительно ускорить операции перемножения больших матриц.

3.1. Последовательная программа

Примем за основу алгоритм перемножения матриц A[M,P]×B[P,N] с накоплением результата в матрице C[M,N]: $\mathbf{C}_{M\times N} \mathrel{+}= \mathbf{A}_{M\times P} \times \mathbf{B}_{P\times N}$, который можно реализовать Си-функцией:

```
void MM(int M, int N, int P,
     float *A, float *B, float *C)
{ int i,j,k; float tmp;
  for (i=0; i<M; i++)
   for (j=0; j<N; j++) { tmp=0.0;
    for (k=0; k<P; k++)
      tmp +=A[i*P+k]*B[k*N+j];
    C[i*N+j]+=tmp;
   }
}//MM
```

в обычной последовательной программе, рассчитанной на исполнение одним процессором.

3.2. Распараллеливание по элементам

Вычисление величины tmp, которая добавляется к элементу C[i,j] в представленном выше алгоритме перемножения матриц, можно производить независимо для каждой пары индексов, поэтому такой алгоритм хорошо поддаётся массовому распараллеливанию.

Представим все имеющиеся рабочие элементы, выделенные для исполнения процедур OpenCL-ядер, в виде двумерного массива M×N. Поручим каждому из них вычислить добавку для элемента общей матрицы C[i,j], расположенной в глобальной памяти, выяснив индексы [i,j] своего расположения в этом массиве путём вызова функции **get_global_id** с аргументом 0 для получения i и 1 – для j. Для этого составим такую процедуру OpenCL-ядра:

```
__kernel void mm_A /* variant A */
(const int M,const int N,const int P,
 __global const float * A,
 __global const float * B,
 __global       float * C) {
 int i= get_global_id(0);
 int j= get_global_id(1);
 int k; float tmp;
 if((i<M)&&(j<N)) { tmp=0.0;
   for(k=0;k<P;k++)
     tmp += A[i*P+k]*B[k*N+j];
   C[i*N+j] += tmp;
 }//if
}// mm_A
```

и оформим её текстовой строкой cSrcString.

Возьмём OpenCL-программу хоста, составленную ранее для векторов, и модифицируем в ней несколько фрагментов.

1. Изменим вызов функции **clCreateKernel**:

```
krnl=clCreateKer-
nel(prgrm,"mm_A",&err);
```

2. Для представления в памяти устройства трёх матриц A,B,C создадим объекты:

```
cl_mem mObA, mObB, mObC;
cl_uint szfl= sizeof(float);
mObA = clCreateBuffer(cntxt,
CL_MEM_READ_ONLY,szfl*M*P,NULL,NULL);
mObB = clCreateBuffer(cntxt,
CL_MEM_READ_ONLY,szfl*P*N,NULL,NULL);
mObC = clCreateBuffer(cntxt,
CL_MEM_READ_WRITE,szfl*M*N,NULL,NULL)
;
```

3. Загрузим буфера этих объектов содержимым матриц A, B, C из основной памяти:

```
err=clEnqueueWriteBuffer(cmndQueue,
mObA,CL_TRUE,0,szfl*M*P,A,0,0,NULL);
err=clEnqueueWriteBuffer(cmndQueue,
mObB,CL_TRUE,0,szfl*P*N,B,0,0,NULL);
err=clEnqueueWriteBuffer(cmndQueue,
mObC,CL_TRUE,0,szfl*M*N,C,0,0,NULL);
```

4. Перед запуском OpenCL-процедуры mm_A назначим ей 6 фактических параметров:

```
cl_uint szui=sizeof(cl_uint);
cl_uint szcm= sizeof(cl_mem);
err=clSetKernelArg(krnl,0,szui,&M);
err=clSetKernelArg(krnl,1,szui,&N);
err=clSetKernelArg(krnl,2,szui,&P);
err=clSetKernel-
Arg(krnl,3,szcm,&mObA);
err=clSetKernel-
Arg(krnl,4,szcm,&mObB);
err=clSetKernel-
Arg(krnl,5,szcm,&mObC);
```

5. Для запуска процедуры ядра на всех рабочих элементах применим такой вызов функции **clEnqueueNDRangeKernel**, который заставляет их работать как двумерный массив M×N :

```
size_t globWS[2]= { M,N };
err=clEnqueueNDRangeKernel(cmndQueue,
krnl,2,NULL,globWS,NULL,0,NULL,&Ev);
```

6. После завершения процедуры ядра прочитаем из буфера объекта mObC в основную память новое значение матрицы C:

```
errNum=clEnqueueReadBuffer(cmnd-
Queue,
mObC,CL_TRUE,0,szfl*M*N,C,0,0,NULL);
```

Такую OpenCL-программу (варианта A), приготовленную для исполнения операции перемножения матриц в среде OpenCL, прогоним многократно с различными размерами M=N=P. Полученные времена исполнения такой операции T_{CL} и $T_{Я}$ сравним с временем исполнения T_{CPU} той же операции функцией MM на одном процессоре и представим в таблице 3.

Таблица 3. Оценка ускорения операции перемножения матриц (варианта A)

MNP	T_{CPU}	T_{CL}	$T_{Я}$	K1	K2
128	5.2298	3.6163	0.7630	1.446	6.854
256	42.6221	6.9226	5.8618	6.157	7.271
512	517.2054	49.5767	48.3442	10.432	10.698
1024	4211.989	518.696	510.39	8.120	8.252
2048	119631.2	7031.64	6970.158	**17.013**	**17.163**
4096	1070548.	80148.8	79756.59	13.357	13.423

T_{CPU} – время исполнения на CPU
T_{CL} – полное время исполнения в OpenCL
$T_{Я}$ – время исполнения OpenCL-ядра
(все времена даны в миллисекундах)
K1 – коэффициент ускорения = **T_{CPU} / T_{CL}**
K2 – коэффициент ускорения = **T_{CPU} / $T_{Я}$**

Из этой таблицы видно, что уже для матриц размером 128×128 исполнение полной операции в среде OpenCL позволяет её ускорить в 1,5 раза (см. коэффициент K1), а само исполнение процедуры ядра (см. K2) – почти в 7 раз. А для матриц большего размера (2048×2048) можно получить еще более значительное ускорение (в 17 раз!). Заметим также, что с ростом размера матриц коэффициент ускорения K1 постепенно приближается к коэффициенту K2, т.к. затраты на подготовку запуска процедуры ядра в среде OpenCL становятся уже менее значительными по сравнению с самим временем её исполнения.

Чтобы добиться ещё большего ускорения, попробуем другие варианты построения параллельной программы с помощью OpenCL.

3.3. Распараллеливание по строкам

Рассмотренный выше вариант (A) OpenCL-программы был рассчитан на массовое параллельное исполнение процедуры ядра сразу всеми рабочими элементами. Но при огромном количестве таких элементов (для M=N=1024 M×N=2^{20}> 1млн.) их реальное параллельное исполнение вряд ли возможно. Ведь количество обрабатывающих элементов, имеющихся на используемом OpenCL-устройстве, может быть значительно меньше.

В таком случае рабочие элементы будут функционировать на обрабатывающих элементах попеременно примерно так же, как на одном процессоре обеспечивается попеременное исполнение нескольких задач. И чем больше рабочих элементов приходится на один обрабатывающий элемент, тем чаще приходится переключать его между ними. Такие накладные расходы на переключения могут замедлять общее время исполнения OpenCL-процедур ядер.

Попробуем построить вариант (B) OpenCL-программы с меньшим числом рабочих элементов. Возьмём их равным количеству строк (M) матрицы C. А каждому рабочему элементу увеличим нагрузку. Пусть теперь он готовит результат не для одного элемента матрицы C, а для всех элементов вверенной ему (i-й) строки. Для такого варианта составим процедуру ядра с тем же заголовком, но в её тело добавим цикл (по j) для перебора элементов строки:

```
__kernel void mm_B(…) /* variant B */
{int j,k; float tmp;
 int i= get_global_id(0);
 if(i<M) for(j=0;j<N;j++){ tmp=0.0;
   for(k=0;k<P;k++)
     tmp += A[i*P+k]*B[k*N+j];
   C[i*N+j] += tmp;
 }
}// mm_B
```

В OpenCL-программе хоста в вызове функции **clCreateKernel** поменяем имя процедуры:

```
krnl=clCreateKer-
nel(prgrm,"mm_B",&err);
```

И организуем запуск процедуры ядра на одномерном массиве из M рабочих элементов:

```
size_t globWS[1]= { M };
err=clEnqueueNDRangeKernel(cmndQueue,
krnl,1,NULL,globWS,NULL,0,NULL,&Ev);
```

Однако прогоны полученного варианта (B) OpenCL-программы ожидаемого улучшения производительности, увы, не показывают. Хуже того, на некоторых аппаратных конфигурациях коэффициенты ускорения оказываются, как ни странно, даже ниже, чем в варианте A.

Тормозом к дальнейшему улучшению производительности оказывается серьёзный недостаток обоих прежних вариантов, заключающийся в том, что все рабочие элементы обрабатывают данные, размещённые в глобальной памяти OpenCL-устройства, и при частом доступе к ним вынуждены терять время в ожидании, пока другие элементы работают с ними. Узкий доступ к глобальной памяти и является тем «бутылочным горлышком», через которое вынуждены «пробираться» рабочие элементы и которое в итоге замедляет их общую работу. Попытаемся далее улучшать OpenCL-программу, сокращая при исполнении OpenCL-процедур обращения к данным в глобальную память.

Прежде всего, заметим, что процедура mm_B в цикле по j при вычислении нового результата tmp, добавляемого к элементу C[i,j], каждый раз заново прочитывает из глобальной памяти элементы i-ой строки матрицы A. Изменим её, предусмотрев в новом варианте (C) этой процедуры одноразовое копирование i-ой строки матрицы A в локальный вектор (vA), размещаемый во внутренней (private) памяти:

```
#define MAX_P 4096
__kernel void mm_C(…) /* variant C */
{ int j,k;float tmp; float vA[MAX_P];
 int i= get_global_id(0);
 for(k=0;k<P;k++) vA[k]=A[i*P+k];
 for(j=0;j<N;j++){ tmp=0.0;
   for(k=0;k<P;k++)
     tmp += vA[k]*B[k*N+j];
   C[i*N+j] += tmp;
 }
}// mm_C
```

Получим OpenCL-программу варианта C, поменяв в ней лишь имя ядра на mm_C. Прогоняя её многократно, можно убедиться, что она действительно улучшает коэффициенты ускорения операции перемножения матриц, но не во всех случаях (см. итоговую таблицу 4 в п. 3.5). Как и следовало ожидать, копирование строки матрицы из глобальной памяти во внутреннюю повышает производительность операции лишь для матриц очень большого размера (≥1024).

3.4. Применение рабочей группы и использование локальной памяти

Для дальнейшей оптимизации заметим, что при вычислении добавочной величины tmp для j-го элемента C[i,j] своей i-ой строки каждый рабочий элемент вынужден считывать из глобальной памяти j-ый столбец матрицы B. Причём такое считывание он будет делать параллельно со всеми другими рабочими элементами, которые в тот же момент крутятся в цикле по k, исполняя процедуру ядра mm_C. Чтобы сократить количество дублирующих чтений одного и того же столбца, объединим усилия нескольких соседних рабочих элементов, для чего соберём их в одну рабочую группу с единой локальной памятью.

В такой группе очередной j-й столбец, необходимый всем рабочим элементам в начале нового витка цикла по j, можно будет совместными усилиями прочитать из глобальной памяти устройства в локальную память своей группы всего один раз. Далее каждый рабочий элемент группы уже сможет прочитывать элементы требуемого столбца из локальной памяти, а не из глобальной, что позволит в итоге поднять их общую производительность.

Схематично замысел предлагаемой оптимизации демонстрируется на рис.3.

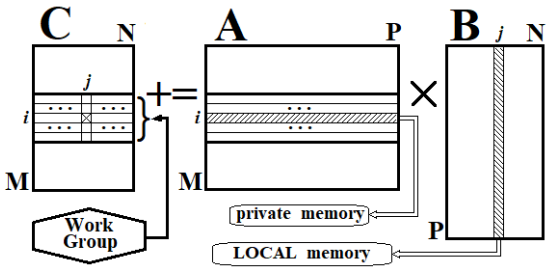


Рис. 3. Схема операции перемножения матриц с применением рабочей группы и локальной памяти

Для её осуществления составим новый вариант (D) OpenCL-процедуры:

```
__kernel void mm_D /* variant D */
(const int M,const int N,const int P,
 __global const float * A,
 __global const float * B,
 __global       float * C,
 __local        float *vB ) {
 int j,k; float tmp; float vA[MAX_P];
 int i= get_global_id(0);
 int il= get_local_id(0);
 int nl= get_local_size(0);
 for(k=0;k<P;k++) vA[k]=A[i*P+k];
 for(j=0;j<N;j++){ tmp=0.0;
  for(k=il;k<P;k=k+nl)vB[k]=B[k*N+j];
  barrier(CLK_LOCAL_MEM_FENCE); /*!*/
  for(k=0;k<P;k++)tmp+=vA[k]*vB[k];
  C[i*N+j] += tmp;
 } }// mm_D
```

В её заголовке новый параметр (vB) с ключевым словом __local задаёт вектор в локальной памяти, куда будет копироваться очередной j-ый столбец матрицы B. Такое копирование осуществляется в начале нового шага цикла по j совместными усилиями всех рабочих элементов группы. Так что рабочий элемент с локальным номером **il** копирует из глобальной памяти в локальную элементы j-го столбца матрицы B вида B[j,k], где k принимает значения ряда арифметической прогрессии с шагом **nl**, начиная с **il**, а **nl** – размер рабочей группы, т.е. количество рабочих элементов в ней.

Действия рабочих элементов группы должны быть синхронизированы так, чтобы они все смогли начать свой цикл по k лишь после того, как они совместными усилиями завершат копирование j-го столбца матрицы B в вектор vB. Для такой синхронизации применяется вызов функции **barrier**. Каждый рабочий элемент приостановит своё исполнение в точке вызова этой функции, чтобы дождаться момента, когда такой же вызов выполнят все остальные рабочие элементы группы. И лишь после этого они смогут продолжать каждый свою дальнейшую деятельность.

Подкорректируем OpenCL-программу хоста для варианта D. Для этого в вызове функции

clCreateKernel поменяем имя процедуры ядра:

```
krnl=clCreateKer-
nel(prgrm,"mm_D",&err);
```

Зададим в качестве характеристики 7-го параметра (vB) процедуры ядра размер (P) вектора вещественных чисел, не уточняя, где именно он будет располагаться в локальной памяти:

```
err=clSetKernel-
Arg(krnl,6,szfl*P,NULL);
```

И организуем запуск процедуры ядра на одномерном массиве из M рабочих элементов, задав размер локальной рабочей группы (**L**) таким, чтобы величина (M/L), определяющая количество рабочих групп, оказалась целочисленной:

```
#define L 128
size_t globWS[1]= { M };
size_t locWS[1]= { L };
err=clEnqueueNDRangeKernel(cmndQueue,
krnl,1,NULL,globWS,locWS,0,NULL,&Ev);
```

Вариант (D) OpenCL-программы, полученный в результате проведённой модификации, оказывается наилучшим по производительности (среди всех рассмотренных) для перемножения матриц больших размеров. Это подтверждается многократными тестовыми прогонами различных вариантов OpenCL-программы, результаты которых представлены в итоговой сравнительной таблице 4 (см. п.3.5).

3.5. Итоговая сравнительная таблица

Результаты оптимизации операции перемножения матриц путём разработки различных вариантов параллельных программ по технологии OpenCL представлены в таблице 4.

Таблица 4. Ускорение операции перемножения матриц разными вариантами OpenCL-программы

вариант		128	256	512	1024	2048	4096
A	**K1**	1.65	6.15	10.43	8.12	**17.01**	13.36
	K2	**6.85**	**7.27**	**10.67**	8.25	**17.16**	13.42
B	**K1**	1.38	3.06	6.17	6.13	5.29	0.82
	K2	1.59	3.19	6.24	6.26	5.32	5.89
C	**K1**	1.00	2.69	6.34	10.68	**62.35**	**57.50**
	K2	1.69	3.48	6.86	11.11	**63.45**	**58.47**
D	**K1**	1.84	4.36	8.10	**15.27**	**72.44**	**58.55**
	K2	2.26	4.66	8.23	**15.65**	**73.2**	**59.24**

K1 – коэффициент ускорения всей операции
K2 – коэффициент ускорения процедуры ядра

Из этой таблицы видно, что для матриц небольшого размера (128,256,512) самый простой вариант (A) распараллеливания по элементам оказывается и самым выгодным, т.к. обеспечивает наибольшие коэффициенты ускорения.

А для матриц большего размера (1024, 2048, 4096) более высокие коэффициенты ускорения обеспечивают варианты (C,D) распараллеливания по строкам с копированием отдельных строк и столбцов перемножаемых матриц во внутреннюю память рабочих элементов и в локальную память рабочих групп, организуемых для совместной согласованной работы нескольких рабочих элементов.

При самом оптимальном варианте (D) построения OpenCL-программы удаётся ускорить операцию перемножения матриц почти в 73 раза (на матрицах размером 2048×2048). Но заметим, что такой высокий коэффициент ускорения рассчитан по отношению к времени исполнения обычной Си-функции (MM в п.3.1) перемножения матриц на одном процессоре.

Однако, с учётом иерархического строения памяти оказывается, что более высокую производительность могут обеспечить другие варианты этой Си-функций перемножения матриц, оптимизированные для исполнения на процессоре с быстродействующей кэш-памятью.

Рассмотрим три варианта модификации функции MM, тела которых различаются лишь порядком вложенности циклов по i, j, k:

```
void MMijk(...) { int i,j,k;
 for (i = 0; i < M; i++)
 for (j = 0; j < N; j++)
 for (k = 0; k < P; k++)
 C[i*N+j]+=A[i*P+k]*B[k*N+j];
}//Mmijk
```

```
void MMkij(...) { int i,j,k;
 for (k = 0; k < P; k++)
 for (i = 0; i < M; i++)
 for (j = 0; j < N; j++)
 C[i*N+j]+=A[i*P+k]*B[k*N+j];
}//Mmkij
```

```
void MMikj(...) { int i,j,k;
 for (i = 0; i < M; i++)
 for (k = 0; k < P; k++)
 for (j = 0; j < N; j++)
 C[i*N+j]+=A[i*P+k]*B[k*N+j];
}//MMikj
```

Все они исполняют ту же самую операцию перемножения матриц, что и функция MM, но с заметно разной длительностью (см. таблицу 5).

Таблица 5. Длительность операции перемножения матриц (в млсек.) у разных вариантов функции MM

MNP	MM	MMijk	MMkij	MMikj
128	5.2072	7.6617	**6.6491**	6.7277
256	45.2643	65.6592	51.2951	**51.0573**
512	516.2746	768.3252	**405.377**	406.9206
1024	5091.67	7479.138	3241.037	**3227.516**
2048	110506.5	154138.8	26099.82	**25880.29**
4096	1029104.1	1456107.	207350.2	**205807.4**

Заметим, что вариант функции MMikj для всех больших матриц работает быстрее, а для матриц размером 2048 и 4096 он превосходит прежний вариант MM уже в несколько раз.

Пересчитаем теперь (для вариантов A,C,D), какие коэффициенты ускорения обеспечивает параллельная программа, разработанная с применением OpenCL, по отношению к самому оптимальному варианту (MMikj) исполнения

функции перемножения матриц на одном процессоре. И представим их в таблице 6.

Таблица 6. Итоговые коэффициенты ускорения при перемножении матриц с применением OpenCL

вариант		128	256	512	1024	2048	4096
A	**K1**	1.854	7.433	8.19	6.16	**3.66**	2.56
	K2	8.785	8.778	8.39	6.26	**3.69**	2.57
C	**K1**	1.285	3.245	4.98	8.10	**13.40**	11.01
	K2	2.164	4.199	5.38	8.43	**13.64**	11.19
D	**K1**	2.363	5.270	6.35	11.58	**15.57**	11.21
	K2	2.893	5.624	6.46	11.87	**15.74**	11.35

Таким образом, с помощью применения технологии OpenCL можно реально ускорить операцию перемножения больших матриц (размером 1024, 2048) примерно в 12-16 раз.

4. Результаты ускорения в среде OpenCL платформы КОМДИВ

Представленные выше показатели ускорения были получены путем прогонов разработанных OpenCL-программ на аппаратной платформе со встроенным графическим процессором (именно такая аппаратная конфигурация и была описана в п. 2.9). На самом деле, с применением технологии OpenCL можно добиться ещё большего ускорения вычислительных операций, если подключить к компьютеру более мощную специализированную видеокарту.

Например, для аппаратной конфигурации с универсальным процессором Intel i3-2100 (на частоте 3.1 Ггц) и подключённой к нему видеокартой NVidia GeForce 1050ti (1392 Мгц) представленные ранее OpenCL-программы выдают более значимые показатели ускорений для тех же вычислительных операций над векторами и матрицами (см. таблицы 7-9).

Таблица 7. Оценка ускорения операции AXPY в среде OpenCL на платформе Intel-NVidia

N	T_{CPU}	T_{CL}	$T_{Я}$	K1	K2
10^3	0.003	0.41702	0.003676	0.007	0.816
10^4	0.040002	0.38002	0.004816	0.105	8.306
10^5	0.4000	0.7500	0.017707	0.533	22.591
10^6	4.1002	5.3003	1.367232	0.774	25.644
10^7	48.503	50.503	1.364608	0.960	35.543

Таблица 8. Оценка ускорения операции SINCOS в среде OpenCL на платформе Intel-NVidia

N	T_{CPU}	T_{CL}	$T_{Я}$	K1	K2
10^3	0.063003	0.35702	0.003687	0.176	17.09
10^4	0.695040	0.33502	0.004304	2.075	161.49
10^5	7.55043	0.95006	0.012635	7.95	597.58
10^6	102.1058	4.50026	0.231676	22.69	440.73
10^7	1155.566	57.0032	2.560592	20.27	451.29

Таблица 9. Ускорение операции перемножения матриц в среде OpenCL на платформе Intel-NVidia

вариант		128	256	512	1024	2048
A	**K1**	5.0	21.00	28.318	29.006	**32.750**
	K2	39.86	46.506	35.380	30.390	**33.346**
C	**K1**	10.0	28.000	32.789	74.826	**92.104**
	K2	21.12	41.988	36.781	83.129	**96.158**
D	**K1**	10.0	28.001	51.917	63.741	**95.695**
	K2	26.04	58.712	58.246	68.980	**99.915**

Выясним теперь, какие же ускорения вычислительных операций можно получить в среде OpenCL, обеспечиваемой в настоящее время микросхемой графического ускорителя семейства КОМДИВ.

Представленные ранее OpenCL-программы прогонялись на этой микросхеме с частотой процессора 400 Мгц и частотой видеокарты 200 Мгц. Полученные в результате этих прогонов коэффициенты ускорения операций обработки векторов и матриц представлены в итоговых таблицах 10-12.

Таблица 10. Оценка ускорения операции AXPY в среде OpenCL на платформе КОМДИВ

N	T_{CPU}	T_{CL}	$T_{Я}$	K1	K2
10^3	0.100585	3.07291	0.54500	0.033	0.1846
10^4	1.033373	3.48531	0.52800	0.296	1.957
10^5	10.59840	9.88295	1.08500	1.072	5.779
10^6	106.6449	76.7783	7.5030	1.389	14.214
10^7	1066.229	899.828	78.410	1.185	13.598

Таблица 11. Оценка ускорения операции SINCOS в среде OpenCL на платформе КОМДИВ

N	T_{CPU}	T_{CL}	$T_{Я}$	K1	K2
10^3	11.8044	3.01745	0.5990	3.912	19.707
10^4	12.13999	11.9686	0.6840	3.598	17.749
10^5	121.5864	2.9217	3.0710	10.159	23.123
10^6	1218.085	94.1239	27.1950	12.941	44.791
10^7	15207.84	1754.44	949.252	8.6682	16.021

Таблица 12. Ускорение операции перемножения матриц в среде OpenCL на платформе КОМДИВ

вариант		512	1024	2048	4096
A	**K1**	2.603	2.566	**2.535**	2.512
	K2	2.612	2.570	**2.537**	2.513
C	**K1**	**4.844**	**4.200**	**2.371**	2.296
	K2	**4.872**	**4.212**	**2.373**	2.297
D	**K1**	**5.019**	**4.547**	**2.431**	2.332
	K2	**5.050**	**4.560**	**2.433**	2.332

Конечно, по сравнению с предыдущими двумя платформами эти результаты выглядят достаточно скромными. Но ведь и получены они на минимально возможной аппаратной конфигурации, обеспечивающей поддержку технологии OpenCL в микропроцессорах семейства КОМДИВ.

Анализируя коэффициенты ускорения, представленные в таблицах 10-12 (и сравнивая их с таблицами 15-20 из [3]), можно заметить, что микросхема графического ускорителя текущей версии обеспечивает почти такие же возможности ускорения вычислительных операций обработки векторов и матриц больших размеров, которые векторный сопроцессор семейства

КОМДИВ обеспечивает лишь для векторов и матриц меньшего размера.

5. Заключение

Полученный автором опыт разработки программ по технологии OpenCL позволяет сделать вывод, что такую технологию можно успешно применять для ускорения тех вычислительных задач, при решении которых требуется исполнять однотипные операции над огромными массивами данных.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме №0065-2019-0004.

Optimization of Matrix Multiplication Operation Based on OpenCL Technology

A.A. Burtsev

Abstract. The article provides an initial introduction to OpenCL technology, which allows to use powerful resources of graphics processors to improve the speed of computing programs. Typical techniques of development in the environment of OpenCL efficient parallel programs for processing vectors and matrices are considered. The main focus is on analyzing the various ways of constructing a program to optimize the operation of multiplication of large matrices.

Keywords: parallel programming, OpenCL technology, heterogeneous systems, microprocessors of the KOMDIV family, matrix multiplication operation

Литература

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

2. С.В. Борзунов, С.Д. Кургалин, А.В. Флегель. Практикум по параллельному программированию. Спб., БХВ, 2017.

3. А.А. Бурцев. О возможности оптимизации некоторых функций библиотеки линейной алгебры с помощью векторного сопроцессора. «Труды НИИСИ РАН», Т. 4 (2014), № 2, 5–15.

4. А.А. Бурцев. О возможности применения векторного сопроцессора для ускорения операции быстрого преобразования Фурье. «Труды НИИСИ РАН», Т. 5 (2015), № 2, 138–147.

5. О.Ю. Сударева. Эффективная реализация алгоритмов быстрого преобразования Фурье и свёртки на микропроцессоре КОМДИВ128-РИО. М., НИИСИ РАН, 2014.

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

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

Сравнение стратегий распараллеливания векторизованного римановского решателя с помощью OpenMP для микропроцессора Intel Xeon Phi KNL

М.Ю. Воробьев[1], А.А. Рыбаков[2], А.Д. Чопорняк[3]

[1]МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия, nordmike@jscc.ru;

[2]МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия, rybakov@jscc.ru;

[3]МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия, adc@jscc.ru

Аннотация. Римановские решатели широко используются в численных методах, при решении задач газовой динамики. При этом во время проведения вычислений требуется решать задачу Римана о распаде произвольного разрыва на каждой итерации расчетов для каждой пары соседних ячеек расчетной сетки. Таким образом, требуется иметь эффективную реализацию римановских решателей. В данной статье рассматривается задача о распараллеливании с помощью OpenMP применения векторизованного точного римановского решателя к массивам входных данных (массивы газодинамических параметров слева и справа от разрыва). Рассматриваются три различные стратегии распараллеливания. Рассматриваемые стратегии распараллеливания были реализованы в программном коде и протестированы на 72-ядерных микропроцессорах Intel Xeon Phi KNL. В результате проведенных численных экспериментов для наиболее эффективной стратегии распараллеливания было получено суммарное ускорение векторизованного римановского решателя в 368 раз (при использовании 139 потоков) на одном микропроцессоре Intel Xeon Phi KNL по сравнению с однопоточной невекторизованной версией решателя.

Ключевые слова: газовая динамика, римановский решатель, векторизация, AVX-512, распараллеливание, OpenMP, Intel Xeon Phi KNL

1. Введение

Точное или приближенное решение задачи Римана о распаде произвольного разрыва используется в численных методах для нестационарных задач с большими разрывами. На решении задачи Римана базируется метод Годунова решения систем нестационарных уравнений газовой динамики [1,2]. В методе Годунова на каждой итерации расчетов на каждой грани между соседними расчетными ячейками решается задача Римана для определения потоков через эту грань. Так как современные расчетные сетки могут содержать десятки миллионов ячеек и более, то для эффективного использования данного метода требуется иметь эффективную реализацию римановских решателей. В мире известно большое количество работ, посвященных оптимизации работы приближенных римановских решателей с помощью векторизации, а также распараллеливания при использовании OpenMP и MPI [3-6]. В частности в работе [5] описаны подходы, позволившие достичь ускорения приближенного римановского решателя за счет векторизации вычислений в диапазоне 2,5-6,5 раз для вещественных операций с двойной точностью.

Данная статья посвящена точному римановскому решателю, который одновременно обладает компактным вычислительным ядром, а с другой стороны, содержит достаточно сложный программный контекст, который не может быть автоматический векторизован современными оптимизирующими компиляторами. В числе особенностей программного кода точного римановского решателя можно отметить сложное управление (с уровнем вложенности условных конструкций, достигающим значения 4), циклы с нерегулярным количеством итераций и вызовы функций.

Вопросы векторизации точного римановского решателя для микропроцессоров Intel Xeon Phi KNL в одномерном случае подробно освещены в работе [7]. При этом для векторизации программного кода использовался набор AVX-512 [8] широких векторных инструкций. Инструкции набора AVX-512 встраивались в программный код с помощью специальных функций интринсиков [9]. Для векторизации сложного программного контекста применялся подход, основный на переводе программного кода в предикатное представление и замене скалярных инструкций на векторные аналоги, описанный в работах [10-12].

2. Описание римановского решателя

В статье рассматривается реализация точного римановского решателя [13], которая находится в свободном доступе в сети Интернет в составе библиотеки NUMERICA [14]. На первом этапе выполнялась векторизация одномерной версии решателя [7]. Рассмотрим краткое описание процесса векторизации. Изначально одномерный римановский решатель по значениям газодинамических параметров (плотность, скорость и давление) справа и слева от разрыва находит значение данных параметров на самом разрыве в нулевой момент времени после устранения перегородки. Это можно записать в виде формулы в следующем виде:

$$U_l = \begin{pmatrix} d_l \\ u_l \\ p_l \end{pmatrix}, \quad U_r = \begin{pmatrix} d_r \\ u_r \\ p_r \end{pmatrix}$$

$$U = \begin{pmatrix} d \\ u \\ p \end{pmatrix} = riemann(U_l, U_r)$$

В данных формулах буквами d обозначены соответствующие значения плотности, u – скорости, p – плотности слева, справа и на самом разрыве соответственно.

Для выполнения векторизации функция, принимающая скалярные входные параметры, была заменена на функцию с векторными входными параметрами следующим образом:

$$\overline{U}_l = \begin{pmatrix} \overline{d}_l \\ \overline{u}_l \\ \overline{p}_l \end{pmatrix}, \quad \overline{U}_r = \begin{pmatrix} \overline{d}_r \\ \overline{u}_r \\ \overline{p}_r \end{pmatrix}$$

$$\overline{U} = \begin{pmatrix} \overline{d} \\ \overline{u} \\ \overline{p} \end{pmatrix} = riemann(\overline{U}_l, \overline{U}_r)$$

После этого массивы входных данных были разделены на отдельные участки длины 16, которые при условии использования вещественных данных одинарной точности упаковываются в векторные регистры _m512. Таким образом, при описанной перекомпоновке входных данных с помощью векторизации стало возможным решать одновременно 16 экземпляров задачи Римана на одном процессорном ядре.

После выполнения полной векторизации кода римановского решателя было достигнуто ускорение 7 раз для одномерного случая [7] и в районе 6,7 раз для решателя, расширенного на трехмерный случай (для этого требуются минимальные изменения по добавлению еще двух составляющих скорости).

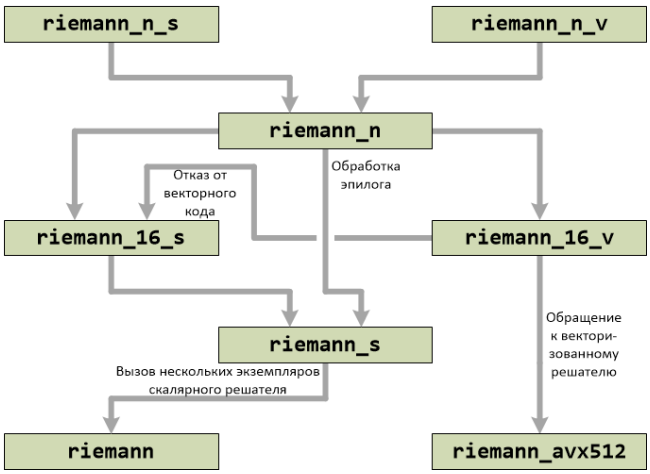


Рис. 1. Схема взаимодействия функций в реализации римановского решателя с поддержкой скалярной и векторизованной версий

Полная реализация векторизованного точного римановского решателя в трехмерном случае доступна в сети Интернет [15].

В числе основных функций в составе римановского решателя можно отметить следующие (см. рис. 1):

– riemann – оригинальная реализация решателя для скалярного набора входных данных.

– riemann_s – скалярная реализация для решения нескольких экземпляров задачи Римана.

– riemann_avx512 – векторная реализация с использованием инструкций AVX-512 решения 16 экземпляров задачи Римана.

– riemann_16_s – скалярная реализация решения 16 экземпляров задачи.

– riemann_16_v – реализация решения 16 экземпляров задачи, данная функция обращается либо к скалярной (riemann_16_s), либо к векторизованной (riemann_avx512) реализации в зависимости от флагов сборки.

– riemann_n – общий вызов функции для решения n экземпляров задачи, в которой можно выбирать, с помощью скалярного или векторного кода должна быть решена задача. Внутри данной функции выполняется реализация стратегий распараллеливания с помощью OpenMP. В частности выполняется разделение массивов входных данных на части по 16 элементов, обращение к функциям riemann_16_s и riemann_16_v, а также обработка эпилога цикла (если длина массивов входных данных не кратна 16, эпилог цикла обрабатывается с помощью функции riemann_s).

– riemann_n_s – решение n экземпляров задачи с помощью скалярного кода.

– riemann_n_v – решение n экземпляров задачи с помощью векторного кода.

3. Определение стратегий распараллеливания с помощью OpenMP

Микропроцессоры Intel Xeon Phi KNL 7290, для которых проводилось исследование по распараллеливанию, содержат по 72 ядра, в каждом из которых возможно запустить по 4 потока, что дает суммарно 288 потоков для одного процессора. Ввиду этого применение распараллеливания с помощью OpenMP является ожидаемым, так как способно существенно ускорить исполняемый код. Были проанализированы 3 стратегии распараллеливания (см. рис. 2), описание которых приведено ниже.

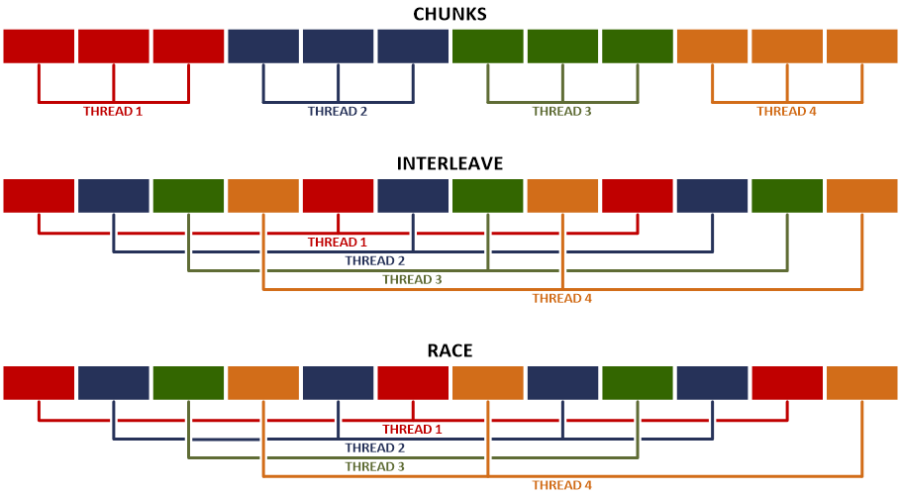


Рис. 2. Иллюстрация работы трех рассматриваемых стратегий распараллеливания вычислений: CHUNKS, INTERLEAVE, RACE

В качестве первой стратегии распараллеливания была использована стратегия CHUNKS, в которой массивы входных данных были разделены на равные непрерывные части по числу используемых потоков, каждый поток обрабатывал свои участки массивов входных данных (на листинге ниже nt – общее количество потоков, solver_16 – указатель на решатель для обработки 16 экземпляров задачи Римана).

```
// [15], riemann.cpp, 585-599
#pragma omp parallel
{
    int tn = omp_get_thread_num();
    int lb = (int)((c / FP16_VECTOR_SIZE) * ((double)tn / (double)nt));
    int ub = (int)((c / FP16_VECTOR_SIZE) * ((double)(tn + 1) / (double)nt));

    for (int i = lb * FP16_VECTOR_SIZE;
         i < ub * FP16_VECTOR_SIZE;
         i += FP16_VECTOR_SIZE)
    {
        solver_16(dl + i, ul + i, vl + i, wl + i, pl + i,
                  dr + i, ur + i, vr + i, wr + i, pr + i,
                  d + i, u + i, v + i, w + i, p + i);
    }
}
```

Во второй стратегии – INTERLEAVE – массивы входных данных были разделены на участки по 16 элементов и распределялись между потоками в шахматном порядке (на листинге ниже c_base – длина массивов входных данных без учета эпилога цикла, nt и solver_16 имеют тот же смысл, что и в листинге выше).

```
// [15], riemann.cpp, 603-615
#pragma omp parallel
{
    int tn = omp_get_thread_num();

    for (int i = tn * FP16_VECTOR_SIZE;
         i < c_base;
         i += nt * FP16_VECTOR_SIZE)
    {
        solver_16(dl + i, ul + i, vl + i, wl + i, pl + i,
                  dr + i, ur + i, vr + i, wr + i, pr + i,
                  d + i, u + i, v + i, w + i, p + i);
    }
}
```

Третья стратегия – RACE – основана на ведении глобального адреса следующего готового к обработке участка входных данных. Как только очередной поток освобождается, он приступает к обработке следующих свободных 16 экземпляров задачи. Таким образом была предпринята попытка избавиться от простоев потоков в случае разного времени выполнения отдельных экземпляров задачи.

```
// [15], riemann.cpp, 620-655
int g = 0;
#pragma omp parallel
{
    int i = 0;
    bool is_break = false;

    while (true)
    {
        #pragma omp critical
        {
            if (g >= c_base)
            {
                is_break = true;
            }
            else
            {
                i = g;
                g += FP16_VECTOR_SIZE;
            }
        }

        if (is_break)
        {
            break;
        }

        solver_16(dl + i, ul + i, vl + i, wl + i, pl + i,
                  dr + i, ur + i, vr + i, wr + i, pr + i,
                  d + i, u + i, v + i, w + i, p + i);
    }
}
```

На данном листинге g – глобальный счетчик следующей свободной партии экземпляров задачи Римана, доступный всем потокам. Для его проверки и продвижения требуется блокировка.

Для всех описанных стратегий были выполнены тестовые расчеты на суперкомпьютере МВС-10П.

4. Выполнение расчетов на суперкомпьютере

Тестирование различных стратегий распараллеливания обработки задачи Римана выполнялось на пакете [15]. Для сборки исполняемого файла под микропроцессор Intel Xeon Phi KNL использовалась строка компиляции следующего вида:

```
mpicc -I src test/*.cpp
src/*.cpp -DINTEL -
O3 -xmic-avx512 -qopt-re-
port=5 -DOPENMP_CHUNKS -DTEST_MODE=1 -
DREPEATS=3 -DINNER_REPEATS=100 -o rie-
mann.out -lm -fopenmp
```

При компиляции необходимо явно включить опцию использования инструкций AVX-512 (-xmic-avx512), также включен максимальный уровень оптимизации -O3. Данная строка предназначена для сборки исполняемого файла для стратегии распараллеливания CHUNKS. Для стратегий INTERLEAVE и RACE необходимо использовать флаги -DOPENMP_INTERLEAVE и -DOPENMP_RACE. Опция -DINNER_REPEATS=100 регулирует количество повторений тестовых сценариев при проведении вычислений (чем выше это количество, тем ниже погрешность замеров времени выполнения). Опция -DREPEATS=3 регулирует количество повторений, выполняемых для замеров времени исполнения теста.

После сборки теста его запуск выполняется с помощью команды

```
./riemann.out 288
```

Единственным параметров теста является максимальное количество потоков, на которых нужно выполнять прогон. При этом сначала тесты прогоняются на невекторизованном решателе с использованием одного потока (относительно этого запуска считается суммарное ускорение решателя). После этого выполняется запуск векторизованного решателя для различного количества потоков, начиная от 1 и заканчивая 288.

По результатам прогона выдается время работы как невекторизованного решателя на одном потоке, так и векторизованной версии для разного количества потоков. Отдельно выполнялись прогоны для стратегий распараллеливания CHUNKS, INTERELEAVE и RACE. По результатам прогонов были построены графики суммарного ускорения векторизованного и распараллеленного римановского решателя для всех трех стратегий, данные графики представлены на рис. 3.

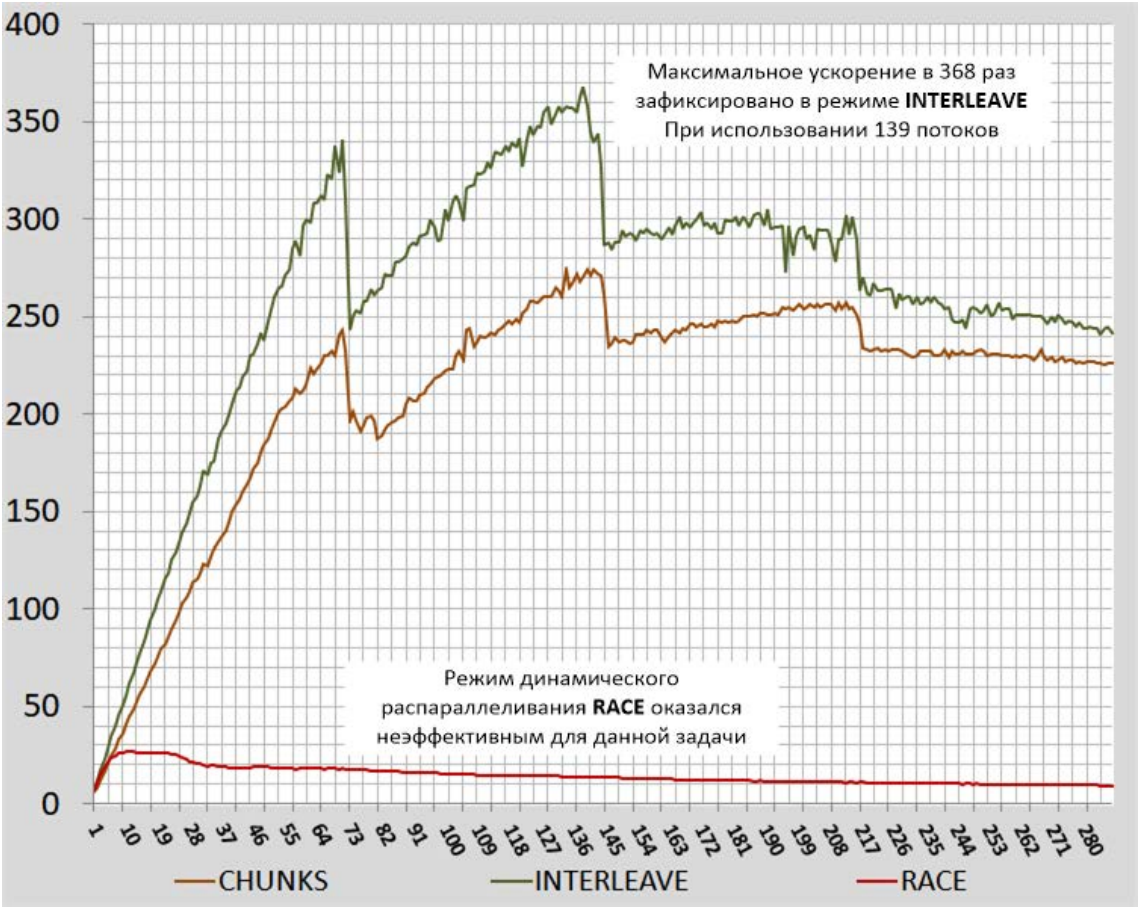


Рис. 3. График ускорения распараллеленного векторизованного римановского решателя по сравнению с нераспараллеленной невекторизованной версией

Из рис. 3 видно, что стратегия распараллеливания RACE является нежизнеспособной даже на сравнительно небольшом числе потоков. Блокировка глобального ресурса (счетчик следующей свободной партии задач), оказывается фатальной и приводит к простою потоков.

Эффективность двух других стратегий примерно одинакова, однако стратегия INTERLEAVE показала себя все же лучше, продемонстрировав максимальное ускорение 368 раз на 139 потоках.

На графиках явно просматриваются 4 характерные участка, длина которых совпадает с количеством ядер микропроцессора Intel Xeon Phi KNL 7290 (72 ядра). На первом участке масштабируемость распараллеливания близка к линейной. На втором участке наблюдается сначала некоторое снижение эффективности, но затем при дальнейшем увеличении количества потоков удается добиться дополнительного ускорения. На третьем и четвертом участках графиков наблюдается снижение производительности. Это ожидаемо, так как каждое ядро микропроцессора содержит два векторных исполнительных устройства и при запуске на ядре трех или четырех потоков начинается конкуренция за данные исполнительные устройства.

5. Заключение

Проведенные исследования по векторизации римановского решателя с помощью набора инструкций AVX-512 и распараллеливанию его с помощью OpenMP показали важность данной задачи, так как комбинирование данных методов позволило добиться ускорения решателя более чем на 2 порядка. Максимальное ускорение по распараллеливанию векторизованного трехмерного решателя составило 368 раз при использовании 139 потоков на микропроцессоре Intel Xeon Phi KNL.

Публикация выполнена в рамках проекта РФФИ «Векторизация римановского решателя для мультиядерных микропроцессоров» (№ 18-07-00638). В работе был использован суперкомпьютер МВС-10П, находящийся в МСЦ РАН.

Comparison of Strategies for Parallelizing a Vectorized Riemann Solver Using OpenMP for the Intel Xeon Phi KNL Microprocessor

M.Yu. Vorobyev, A.A. Rybakov, A.D. Chopornyak

Abstract. Riemann solvers are widely used in numerical methods, in the solution of gas dynamics problems. During the calculations, it is required to solve the Riemann problem on the decay of an arbitrary discontinuity at each iteration of the calculations for each pair of neighboring cells of the computational grid. Thus, an efficient implementation of Riemann solvers is required. This article discusses the problem of parallelization using OpenMP of applying a vectorized exact Riemann solver to arrays of input data (arrays of gas dynamic parameters to the left and right of the discontinuity). Three different parallelization strategies are considered. The considered parallelization strategies were implemented in software code and tested on 72-core Intel Xeon Phi KNL microprocessors. As a result of numerical experiments for the most efficient parallelization strategy, the total acceleration of the vectorized Riemann solver by a factor of 368 (using 139 threads) was obtained on one Intel Xeon Phi KNL microprocessor in comparison with the single-threaded unvectorized version of the solver.

Keywords: gas dynamics, Riemann solver, vectorization, AVX-512, parallelization, OpenMP, Intel Xeon Phi KNL

Литература

1. А.Г. Куликовский, Н.В. Погорелов, А.Ю. Семенов. Математические вопросы численного решения гиперболических систем уравнений. ФИЗМАТЛИТ, М., 2001, 608 с.

2. С.К. Годунов, А.В. Забродин, М.Я. Иванов, А.Н. Крайко, Г.П. Прокопов. Численное решение многомерных задач газовой динамики. Наука, М., 1976, 400 с.

3. I. Kulikov, I. Chernykh, V. Vshivkov, V. Prigarin, V. Mironov, A. Tatukov. The parallel hydrodynamic code for astrophysical flow with stellar equation of state. Proceedings of Russian Supercomputing Days 2018, 612-624.

4. M. Bader, A. Breuer, W. Höltz, S. Rettenberger. Vectorization of an augmented Riemann solver for

the shallow water equations. Proceedings of the 2014 International Conference on High Performance Computing and Simulation, HPCS 2014, 2014, 193-201.

5. C.R. Ferreira, K.T. Mandli, M. Bader. Vectorization of Riemann solvers for the single- and multi-layer shallow water equations. Proceedings of the 2018 International Conference on High Performance Computing and Simulation, HPCS 2018, 2018, 415-422.

6. H.-Y. Schive, U.-H. Zhang, T. Chiueh. Directionally unsplit hydrodynamic schemes with hybrid MPI/OpenMP/GPU parallelization in AMR. International Journal of High Performance Computing Applications, 2011, 367-377.

7. A.A. Rybakov, S.S. Shumilin. Vectorization of the Riemann solver using AVX-512 instruction set. Program Systems: Theory and Applications, 2019, Vol. 10, № 3 (42), 41-58.

8. Intel 64 and IA-32 architectures software developer's manual. Combined volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D and 4. Intel Corporation, 2019.

9. Intel Intrinsics Guide. https://software.intel.com/sites/landingpage/IntrinsicsGuide/, дата обращения 31.08.2020.

10. B.M. Shabanov, A.A. Rybakov, S.S. Shumilin, Vectorization of high-performance scientific calculations using AVX-512 instruction set. Lobachevskii Journal of Mathematics, 2019, Vol. 40, No. 5, 580-598.

11. А.А. Рыбаков, С.С. Шумилин. Исследование эффективности векторизации гнезд циклов с нерегулярным числом итераций. Программные системы: Теория и алгоритмы, 2019, Т. 10, № 4 (43), 77-96.

12. А.А. Рыбаков, С.С. Шумилин. Векторизация сильно разветвленного управления с помощью инструкций AVX-512. Труды НИИСИ РАН, 2018, Т. 8, № 4, 114-126.

13. E.F. Toro. Riemann solvers and numerical methods for fluid dynamics. A practical introduction. 2nd edition, Springer, 1999.

14. NUMERICA, A Library of Sources for Teaching, Research and Applications, by E. F. Toro. https://github.com/dasikasunder/NUMERICA, дата обращения 31.08.2020.

15. https://github.com/r-aax/riemann_vec/releases/tag/metric, дата обращения 31.08.2020.

Методы удаленной отладки ПЛК в среде ТСАГ СПО

А.И. Грюнталь[1], К.Г. Нархов[2]

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

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

Аннотация. Статья содержит описание средств отладки программного обеспечения программируемых логических контроллеров, функционирующих под управлением отечественной операционной системы реального семейства Багет. Описывается используемый для отладки оригинальный протокол REST-MLCP, а также методы совместного использование отладчика со средой разработки ТСАГ СПО, предназначенной для разработки приложений реального времени.

Ключевые слова: отладчик, программируемый логический контроллер, операционная система реального времени, протокол REST-MLCP; ТСАГ СПО, среда разработки, формат JSON

1. Введение

В статье рассматриваются особенности удаленной отладки специального программного обеспечения программируемых логических контроллеров (СПО ПЛК) с использованием среды разработки ТСАГ СПО (технологические средства автоматизированной генерации специального программного обеспечения). Предлагаемая технология отладки ориентирована на отладку программного обеспечения АСУ ТП.

Авторы представляют программные решения для визуализации контролируемых технологических параметров (КТП) объекта автоматизации, инструменты трассировки и протоколирования функционирования СПО ПЛК. Приводится спецификация протокола отладки REST-MLCP, а также описывается программный интерфейс библиотеки обмена отладочной информацией между СПО ПЛК и средой ТСАГ СПО.

Описываемые средства удалённой отладки представляют собой развитие технологических средств визуализации КТП, применяемых в ПЛК, функционирующих под управлением операционной системы реального времени семейства Багет (ОС РВ Багет 2.6 [3]).

ОС РВ Багет 2.6 – это отечественная операционная система, предназначенная для создания программного обеспечения вычислительных систем, работающих в режиме реального времени (ВС РВ). В публикациях [5,9] были рассмотрены методы протоколирования и визуализации значений параметров состояния специального программного обеспечения, а также методы анализа и интерпретации получаемых при этом данных.

Для обеспечения удаленной отладки используется разработанная в ФГУ ФНЦ НИИСИ РАН библиотека мониторинга (БМ), содержащая механизмы контроля состояния специального программного обеспечения, сравнения значения параметров состояния СПО с эталонными (прогнозируемыми) значениями параметров, генерации корректирующих воздействий на СПО при отклонении значений параметров состояния от прогнозируемых значений [1,2,5,6].

Библиотека мониторинга обеспечивает контролирование работоспособности, корректности и целостности специального программного обеспечения на основании данных, получаемых от агентов мониторинга исполняемых в контексте СПО и аккумулирующих данные о выполнении прикладной программы [4].

В качестве технологического средства для визуализации и отладки СПО ПЛК используется среда разработки ТСАГ СПО [7,8], исполняемая на инструментальной ЭВМ с микропроцессорной архитектурой Intel с операционной системой ОС Linux Fedora. Также в качестве инструментальной ЭВМ может использоваться ЭВМ Багет Р2А с микропроцессором 1890ВМ8Я, функционирующая под управлением операционной системой Debian.

2. Разработка приложений в среде ТСАГ СПО

Программа ТСАГ СПО является интегрированной средой, предназначенной для разработки специального программного обеспечения вычислительных систем реального времени (ВС РВ), функционирующего под управлением операционной системы ОС РВ Багет 2.6. Программа ТСАГ СПО представляет собой совокупность следующих взаимодействующих программ (подсистем): ППГИ, ПГП, ПТП и ПГЕН. Назначение

этих подсистем приводится ниже.

Программа ППГИ – это подсистема графического интерфейса (графическая оболочка), обеспечивающая в режиме интерактивного графического диалога создание и ведение проектов и запуск в ТСАГ СПО других подсистем. Подсистема ППГИ построена на основе и является развитием среды разработки СКРПРВ [4].

Программа ПГП – это подсистема графического представления, назначение которой – создание и редактирование графического представления ВС РВ в терминах UML-диаграмм. Подсистема ПГП предназначена для визуального проектирования приложений в виде UML-диаграмм.

Программа ПТП – это подсистема текстовой поддержки графического представления. Подсистема предназначена для создания и редактирования XML-файлов (XML-схем и XML-документов), содержащих типовые заготовки конструкций алгоритмического языка Си. Подсистема ПГП применяется для построения диаграмм программной деятельности;

Программа ПГЕН – это подсистема генерации исходных текстов. Программа осуществляет синтаксический анализ и разбор XML-документов и используется для преобразования диаграмм программной деятельности (представленных в виде XML-представлений UML-модели проектируемой системы) в набор файлов с исходными текстами на языке Си.

Графическое моделирование, создание и ведение проектов вычислительных систем обеспечивается подсистемами ППГИ и ПГП. Порядок работы с проектами в ТСАГ СПО подробно изложен в документах [8, 9].

Среда ТСАГ СПО используется для проектирования аппаратной структуры вычислительный системы и структуры программного обеспечения. Вычислительные системы реального времени могут функционировать на *аппаратной платформе*, состоящей из нескольких ЭВМ (целевых ЭВМ), каждая из которых может содержать несколько процессорных модулей. Среда проектирования ТСАГ СПО ориентирована на поддержку таких многомашинных и многопроцессорных систем.

Структура программного обеспечения ВС РВ средствами ТСАГ СПО отображается на структуру *аппаратной платформы* и соответствует функциональной структуре решаемых вычислительной системой задач.

Программное обеспечение ВС РВ состоит из СПО (прикладных программ), операционной системы реального времени и пакетов поддержки модулей, в совокупности обеспечивающих решение возложенных на ВС РВ задач обработки данных и управления.

Пакет поддержки модуля (ППМ) – это программа, поддерживающая программный интерфейс к оборудованию (аппаратной платформе) для операционной системы реального времени и прикладного программного обеспечения.

Прикладные программы представляют собой совокупность программных функций, представленных программными файлами (программными модулями), файлами определений структур данных, переменных и типов данных, общих для всех прикладных программ ВС РВ или для отдельных функциональных групп прикладных программ.

В общем случае программное обеспечение ВС РВ состоит из программного обеспечения входящих в состав вычислительных систем целевых ЭВМ и групп функциональных программ, которые могут использоваться в некоторых или во всех процессорных модулях ВС РВ. Программное обеспечение целевой ЭВМ состоит из программного обеспечения входящих в состав целевой ЭВМ процессорных модулей и общих для целевой ЭВМ групп функциональных программ, которые могут использоваться в некоторых или во всех процессорных модулях целевой ЭВМ.

Программное обеспечение процессорного модуля состоит из собственных прикладных программ, представленных группами программ, сформированными по функциональному принципу (назначению), из групп функциональных программ целевой ЭВМ, общих для ее процессорных модулей, и из групп функциональных программ ВС РВ, общих для всей системы. Кроме указанных программ в состав СПО процессорного модуля входит образ операционной системы, сформированный из указанных прикладных программ, программ, входящих в состав операционной системы, и ППМ, специфических для конкретной архитектуры процессорного модуля.

3. Спецификация протокола удаленной отладки

Удаленная отладка специального программного обеспечения ПЛК выполняется с помощью инструментальной ЭВМ (ИЭВМ) в ходе интерактивного сеанса.

В условиях АСУ ТП данные от ПЛК поступают в сервер ввода-вывода. Сервер ввода вывода (СВВ) представляет собой программно-аппаратный комплекс, предназначенный для буферизации и накопления технологических параметров состояния в базе данных. В штатном режиме функционирования АСУ ТП сервер ввода-вывода используется как средство информаци-

онного взаимодействия ПЛК со SCADA-станцией, используемой для контроля технологического процесса объекта управления.

В режиме отладки СВВ работает в режиме ответов на запросы клиентов («work-on-request»), которыми являются ПЛК (сохранение значений технологических параметров состояния на СВВ) и ИВЭМ (загрузка значений технологических параметров из СВВ). Далее по тексту трактом отладки будем называть находящиеся в одной сети и информационно взаимодействующие по протоколам отладки ПЛК, СВВ и ИЭВМ.

Для взаимодействия ПЛК и ИЭВМ в рамках тракта отладки используется специально разработанный в ФГУ ФНЦ НИИСИ РАН протокол REST-MLCP (Representational State Transfer — «передача состояния представления» — Monitoring Library Communication Protocol — «протокол взаимодействия с библиотекой мониторинга»). Протокол REST-MLCP является прикладной надстройкой над протоколом HTTP (HyperText Transfer Protocol).

Для обмена данными между клиентом и сервером в качестве формата представления ресурсов и данных в протоколе REST-MLCP используется формат обмена данными JSON (Java Script Object Notation). Единицей приёма-передачи данных в формате JSON является JSON-пакет, синтаксис и семантика которого определяется схемой JSON.

Функционал протокола REST-MLCP реализуется посредством клиентской библиотеки, предоставляющей программный интерфейс (API) для доступа к СВВ.

В состав API входят следующие функции:

– `char ** rest_mlcp_get_ctrl_tags( int controller_id)` – получить список всех КТП для ПЛК с заданным controller_id;

– int rest_mlcp_get_value(int controller_id, char * tag) – получить значение КТП для ПЛК с заданным controller_id;

– `int rest_mlcp_get_values( int controller_id, int mode, struct mlcp_tag *)` – получить значения всех КТП СПО с типом `mode` для ПЛК с заданным `controller_id` (`mode` – тип КТП: 0 – доступ на чтение, 1 – доступ на чтение или запись);

– `int rest_mlcp_set_value( int controller_id, char * tag)` – установить значение КТП с именем tag для ПЛК с заданным `controller_id` (устанавливаемый КТП должен поддерживать доступ на запись);

– `int rest_mlcp_set_values( int controller_id, struct mlcp_tag *)` – установить значения всех КТП для ПЛК с заданным `controller_id`, поддерживающих доступ на запись;

– int rest_mlcp_command_request(int controller_id, int comm, int value) – передать команду на ПЛК с заданным controller_id;

– struct mlcp_tags *rest_mlcp_init_tags(unsigned int controller_id, char *tagnames[], int len) – инициализация КТП для ПЛК с заданным controller_id;

– `int rest_mlcp_access_value(char * tag, int mode, int value)` – доступ к КТП (чтение или запись) на ПЛК (`mode` – тип КТП: 0 – доступ на чтение, 1 – доступ на чтение или запись); в режиме записи аргумент `value` должен быть больше или равен нулю;

– `int rest_mlcp_response_callback(void (*f)(struct mlcp_response *))` – зарегистрировать пользовательский обработчик ответа по протоколу REST-MLCP.

Клиентская библиотека REST-MLCP написана на языке Си и является кроссплатформенной в части компиляции, сборки и запуска для Intel и MIPS архитектур. Для отладки клиента REST-MLCP подготовлен Docker-контейнер [10] с прототипом «Сервера ввода-вывода», поэтому отладка может быть выполнена на инструментальной ЭВМ с предустановленным ПО Docker [10] и без дополнительных программно-аппаратных ресурсов или виртуализации.

Принципиальная схема тракта отладки представлена на рисунке 1.

Состав тракта отладки:

– инструментальная ЭВМ: рабочая станция с микропроцессорной архитектурой Intel или ЭВМ Багет Р-2А с микропроцессором 1890ВМ8Я под управлением ОС семейства Linux с предустановленными интегрированной средой разработки ТСАГ СПО и библиотекой поддержки протокола REST-MLCP;

– сервер ввода-вывода: сервер (ЭВМ с микропроцессорной архитектурой Intel с ОС семейства Linux или ЭВМ с микропроцессором 1890ВМ8Я с ОС РВ Багет 3.5). На сервере должна быть предустановлена СУБД и веб-сервер, который предназначен для приема-передачи JSON-пакетов.

– программируемый логический контроллер (ПЛК Багет Р8) с установленными ОСРВ Багет 2.6 со специальным программным обеспечением СВВ, обеспечивающим логику его функционирования. В составе СПО СВВ должна входить библиотека поддержки протокола REST-MLCP.

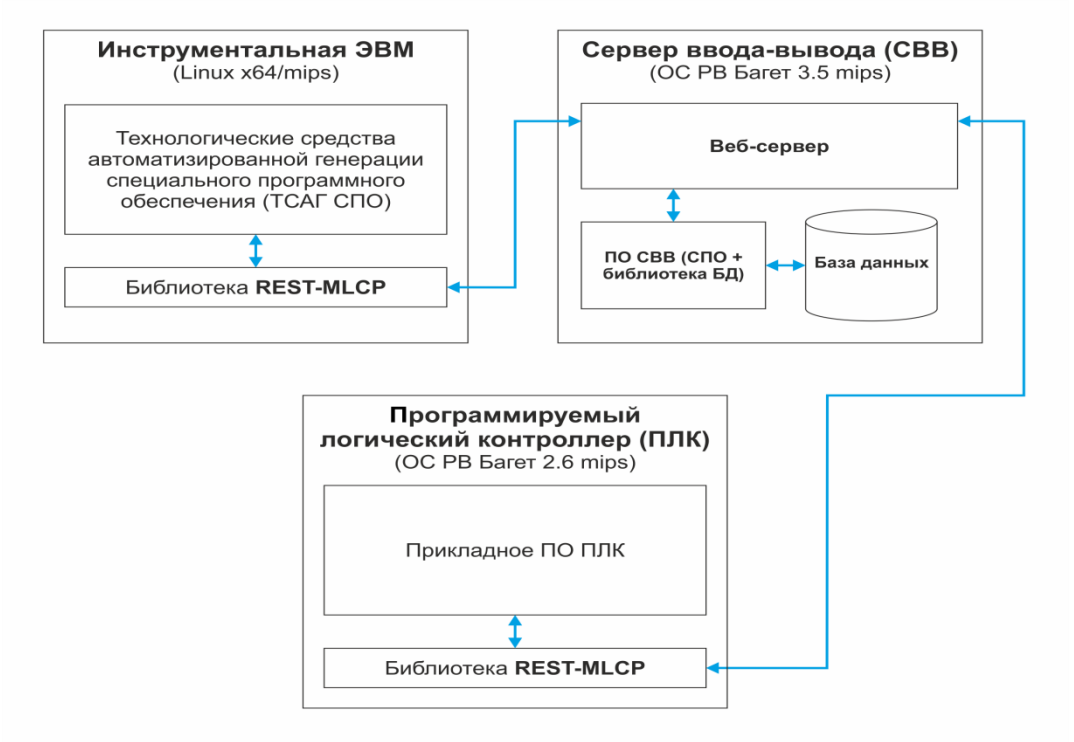


Рис. 1. Тракт отладки ПЛК.

4. Требования к встроенному в ТСАГ СПО отладчику

Встроенный в ТСАГ СПО отладчик соответствует следующим требованиям:

– кроссплатформенность и единство графического пользовательского интерфейса на различных дистрибутивах операционных систем семейства Linux;

– поддержка многопользовательского режима функционирования, обеспечивающая минимизацию конфликтов работы нескольких пользователей;

– интеграция в единое рабочее пространство визуального инспектора отладки и интерфейсных элементов подсистемы ППГИ;

– управление отображением отладочных данных, возможность изменения стилевых и визуальных характеристик. В том числе предусмотрены функции, позволяющие скрывать или показывать таблицы, вкладки, окна и т.п.; создавать иерархические списки объектов отладки, менять цвет объектов и текста; масштабировать как сами объекты и текст, так и рабочую область отладчика в целом;

– сохранение отладочной информации в виде текста в формате XML;

– обеспечение стандартных функций интерфейса, таких как перемещение по рабочему пространству; уменьшение/увеличение масштаба; контекстных меню, наличие диалоговых окон, горизонтальных и вертикальных полос прокрутки; клавиатурный ввод и пр.;

– поддержка специальных окон (редакторов) свойств контролируемых технологических параметров;

– поддержка интерактивного режима работы пользователя с отладчиком (консоль, журнал и т.д.);

– возможность подключения дополнительных модулей (масштабируемость отладчика) для отображения отладочных данных с применением современных средств визуализации, например, OpenGL.

5. Интеграция отладчика в подсистему представления графических интерфейсов

Отладчик должен быть включён в состав исходных текстов подсистемы ППГИ в виде набора программных и информационных модулей. Ниже приводятся необходимые для этого технические сведения.

Исходные тексты отладчика должны быть размещены в каталоге `./sources/AppCore/Debugger`. Для компиляции и сборки исходных текстов отладчика необходимо внести следующие изменения в исходные тексты подсистемы ППГИ.

1) В файле `./sources/config.make` к переменной `INCLUDE_SRC` следует добавить путь `–I $(SRC_DIR)/AppCore/Debugger.`

2) В файле `./sources/AppCore/objects.make` перед строкой `$(EditorsObjectsTMP)` добавить строку `./AppCore/Debugger/*.o \.`

3) В файл `./sources/AppCore/Debugger/Makefile` добавить содержимое в соответствии с листингом, приведённым на рисунке 2.

```
SRC_DIR=../../include
$(SRC_DIR)/config.make

debugger.o: debugger.c debug-
ger.h
```

Рис. 2. Содержимое Makefile подсистемы ППГИ

4) В файле `./bin/ISRP_Data/forms/Menu/main_MenuView.xml` в конец `<widget class="menu" name="menuView">` добавить содержимое в соответствии с листингом, приведенным на рисунке 3.

```
<widget visible="TRUE"
        name="sep_020"
        class="menu_separa-
tor"/>
  <widget
class="check_menu_item"
        name="Debugger"
        value="FALSE">
    <label>
      <string     language="Eng-
lish">
        Debugger
      </string>
      <string     language="Rus-
sian">
        _Отладчик
      </string>
    </label>
    <signal name="activated">
    <action
      name="editor_set_debug-
ger"/>
    </signal>
  </widget>
```

Рис. 3. Содержимое `main_MenuView.xml`

В исходных текстах отладчика должны присутствовать функции `init_debugger()` и `editor_set_debugger()`. В состав исходных текстов отладчика должен входить информационный модуль `debugger.h`, минимальное достаточное содержимое которого представлено на рисунке 4

```
#ifndef DEBUGGER_H
#define DEBUGGER_H
  extern       int       de-
buger_init(void);
#endif /* DEBUGGER_H */6.
```

Рис. 4. Содержимое `debugger.h`

Функция `editor_set_debugger()`должна быть зарегистрирована в init_debugger()с помощью следующего макроса: `GUI_ACTION_REG(editor_set_debugger);`

Функция `init_debugger()` должна быть вызвана в функции-инициализаторе подсистемы ППГИ РВ `app_init(): //Init debugger`

```
#include "debugger.h"
debugger_init();
```

Для доступа к КТП и их визуализации с использованием графического интерфейса ППГИ применяются функции библиотеки поддержки протокола REST-MLCP. Отметим, что помимо собственно поддержки протокола REST-MLCP библиотека поддержки протокола обеспечивает доступ к локальным копиям КТП, гарантируя атомарность модификации КТП СПО и защиту от параллельного доступа при записи.

6. Пользовательский интерфейс отладчика ТСАГ СПО

Пользовательский интерфейс отладчика ТСАГ СПО удовлетворяет требованиям из раздела 4, а также содержит оконно-диалоговые интерфейсные компоненты, обеспечивающие просмотр конфигураций, просмотр списка отлаживаемых переменных и их значений (контролируемых технологических параметров), ввод и вывод данных с эмулятора алфавитно-цифрового терминала, протоколирование, визуализацию переменных в виде графиков.

Запуск отладчика выполняется из подменю «Отладчик» меню «Вид» главного окна ППГИ (рисунок 5). Окно встроенного отладчика КТП разделено на четыре рабочих области:

– загрузчик отладочных конфигураций (отладочная конфигурация содержит общие настройки отладчика, список КТП, допустимые диапазоны значений и т.п.);

– список переменных, заданных

– в формате «Название переменной» - «Значение»;

– область визуализации переменных

– в виде графиков;

– область просмотра конфигурации отладочной сессии, ввода и вывода данных с системной консоли, протоколирования отладочной сессии, отображения ошибок и анализа данных, передаваемых по протоколу REST-MLCP.

Внешний вид окна отладчика приведён на рисунке 6.

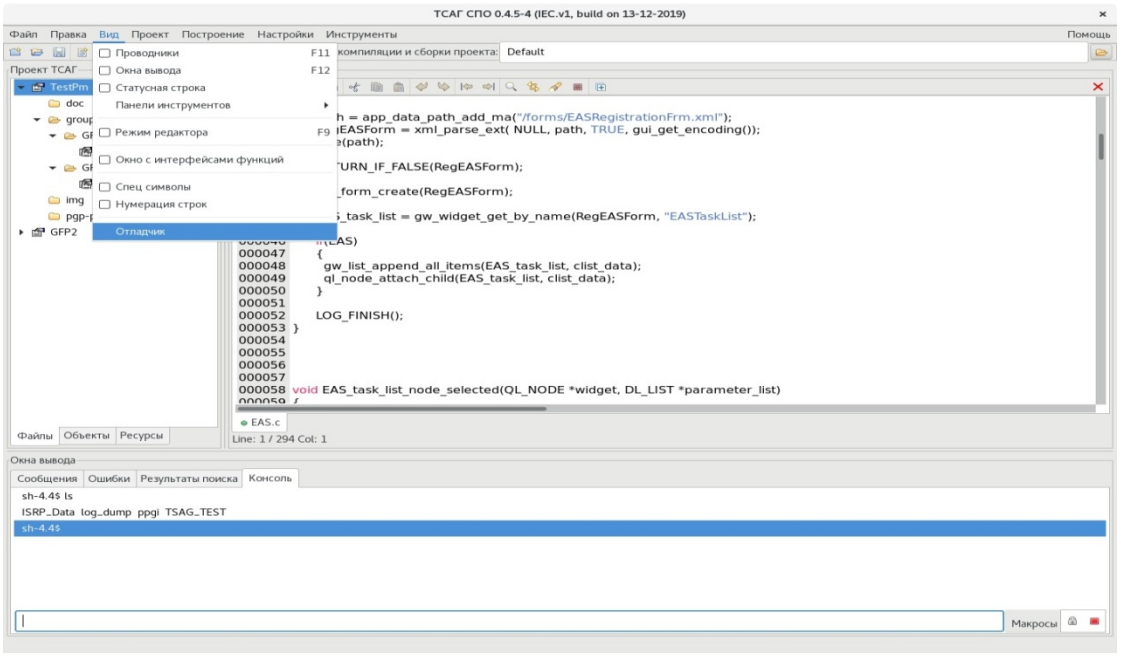


Рис. 5. Меню запуска отладчика в ППГИ

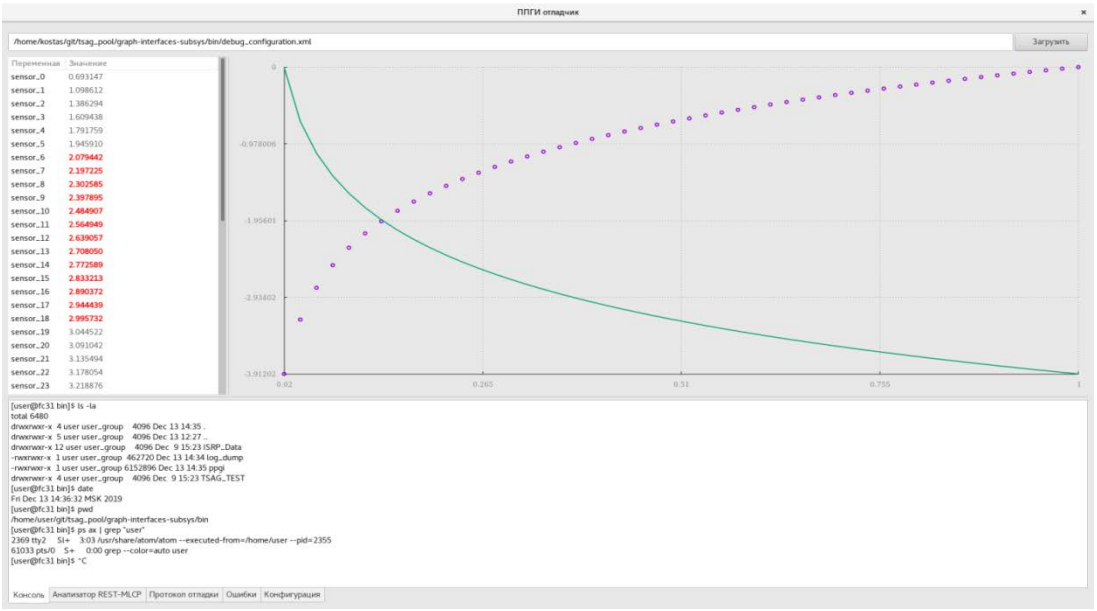


Рис. 6. Внешний вид окна отладчика

7. Заключение

Представленная в статье программно-аппаратная архитектура тракта отладки программного обеспечения ПЛК позволяет использовать библиотеку мониторинга и среду разработки

ТСАГ СПО для отладки и визуализации контролируемых технологических параметров приложений, соответствующих принципу контролируемого выполнения.

Разработана технология, обеспечивающая автоматизацию процесса отладки в интерактивном режиме. Базовый элемент технологии - универсальный протокол передачи данных REST-MLCP, первоначально предназначенный для передачи данных между ПЛК и SCADA- станциями.

Применение технологии приводит к повышению надежности и отказоустойчивости приложений, поскольку гарантирует контролируемое использование примитивов операционной системы в составе приложения. Технология освобождает разработчика от использования консольного отладчика и позволяет исследовать полученные в результате отладки протоколы в сторонних средах, например MatLab, что в свою очередь увеличивает скорость и эффективность отладки СПО.

В статье исследован и описан подход к интеграции библиотеки мониторинга и разработанного ранее технологического пакета ТСАГ СПО. Изложенный подход применим к программируемым логическим контроллерам.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН (проведение фундаментальных научных исследований 47 ГП) по теме № 0065-2019-0002 «Исследование и реализация программной платформы для перспективных многоядерных процессоров» (рег. № АААА-А19-119012290074-2)

Remote PLC Software Debugging Using TSAG SPO Environment

A. Gryuntal, K. Narkhov

Abstract. This paper describes the debugging technology designed to visualize PLC runtime parameters using TSAG SPO development environment. Original data transfer protocol REST-MLCP exploiting JSON format and implementation features are under consideration.

Keywords: debugger, controlled execution, monitor library, real-time operating system, PLC, MLCP-protocol

Литература

1. В.А. Галатенко, К.А. Костюхин, К.А., Н.В. Шмырев Контролируемое выполнение М., НИИСИ РАН, 2012.

2. В.Б. Бетелин, В.А. Галатенко, К.А. Костюхин Контролируемое выполнение с явной моделью, «Программирование», Т 40, (2014), N 5, 45-55.

3. А.Н. Годунов, В.А Солдатов. Операционные системы семейства Багет (сходства, отличия и перспективы) «Программирование», Т 40, (2014) № 5, с. 68-76.

4. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Реализация принципа контролируемого выполнения для прикладных программ реального времени. «Труды НИИСИ РАН», Т 5 (2015) № 2. 113-121.

5. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Библиотека мониторинга для многопоточных программ, «Труды НИИСИ РАН», Т 7 (2017) № 1, 70-74.

6. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Обработка исключительных ситуаций с использованием библиотеки мониторинга, «Труды НИИСИ РАН» Том 7 (2017) № 4, 96-101.

7. К.Г. Нархов, Генератор текста программ в исходном виде для систем реального времени, «Программные продукты и системы». (2010) № 4., 24-30

8. К.Г.Нархов, Методы сокращения количества уязвимостей в специальном программном обеспечении реального времени, «Программные продукты и системы» (2012). № 3. 89-95.

9. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Визуализация параметров состояния прикладной программы на персональной ЭВМ Багет P2A «Труды НИИСИ РАН», Т 8, (2018) № 6, 131-136.

10. Э. Моуэт. Использование Docker. М, ДМК Пресс, 2017.

Построение изображения растровой электронной карты с использованием программных средств для операционной системы реального времени семейства «Багет»

П.В. Егоров [1]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, egorov@niisi.ras.ru, +7(903)5524887.

Аннотация. Рассматривается технология построения изображения растровой электронной карты с использованием программных средств для операционной системы реального времени семейства «Багет». Предлагаемые в статье алгоритмы выполнения операций позволяют достигнуть заданных временных параметров отображения растровой электронной карты на экране дисплея при исполнении контрольной задачи на ЭВМ "Багет-47-51".

Ключевые слова: растровая электронная карта, тайловая карта, тайл, графический сервер, библиотека интерфейсных компонентов, графическая библиотека, библиотека «tilelib».

1. Введение

В статье описывается технология отображения растровой электронной карты на экране дисплея с использованием следующих программных изделий (ПИ), разработанных в ФГУ ФНЦ НИИСИ РАН: библиотека интерфейсных компонентов (БИК), графический сервер (ГС), графическая библиотека, реализующая X – интерфейс, далее графическая библиотека (ГБ), операционная система реального времени семейства «Багет» (ОС РВ).

При разработке технологии требовалось обеспечить заданные временные параметры визуализации геопространственных данных при выполнении контрольной задачи (КЗ) на ЭВМ "Багет-47-51".

Растровой электронной картой будем называть электронную карту геопространственные данные которой представлены в растровой форме, то есть в виде матрицы, элементами которой являются коды цветов картографического изображения [1]. Элемент растра также будем обозначать термином «пиксель». Пиксель - это наименьшая единица растрового изображения. Он характеризуется определённым цветом, яркостью и, возможно, прозрачностью. Один пиксель может хранить информацию только об одном цвете, который и ассоциируется с ним [2].

Одним из преимуществ растрового формата является то, что в настоящее время в большинстве устройств ввода-вывода используется растровый принцип вывода графической информации, поэтому данные в растровом формате могут быть отправлены на такие устройства без дополнительной обработки.

Реализация растровой электронной карты может быть, как в виде одной матрицы пикселей, так и в виде нескольких матриц, которые в процессе отображения «бесшовно» объединяются в одну.

Растровую электронную карту, модель данных которой представлена в виде набора матриц пикселей, будем называть тайловой картой, а саму матрицу пикселей тайлом. В статье будет использоваться термин «тайл», так как он чаще встречается в технической литературе [3, 4, 5].

Технология разработана для модели данных тайловой карты, поэтому далее вместо термина «растровая электронная карта» будет использоваться термин «тайловая карта».

Применение тайловых карт в программах позволяет повысить их производительность путем уменьшения объема обрабатываемых данных и распределения вычислений на несколько процессоров.

Например, структура данных тайловой карты позволяет в программе отображать только те тайлы на экране дисплея, которые требуются в данный момент пользователю, а при сдвиге изображения карты использовать тайлы повторно с помощью метода двойной буферизации.

Свойства тайловых карт, которые требуют соглашения или стандартов, включают размер тайлов, нумерацию уровней масштабирования, использование проекции, способ нумерации или идентификации отдельных тайлов и способ их

запроса. Примерами таких соглашений являются стандарты «Tile Map Service» и «Web Map Tile Service» [6].

В рассматриваемой технологии применяется архитектура программной системы, которая состоит из следующих компонентов:

- операционная система реального времени семейства «Багет»;
- библиотека функций отображения тайловой карты на экране дисплея для языка программирования Си, которая далее будет именоваться «tilelib»;
- библиотека интерфейсных компонентов;
- графический сервер;
- графическая библиотека;
- прикладная программа;
- подпрограмма, реализующая операции с данными тайловой карты, которую будем называть драйвером.

Применение драйвера в программной архитектуре позволяет сделать библиотеку «tilelib» инвариантной к модели данных тайловой карты. Для каждой модели предполагается создание отдельной программы драйвера. Для библиотеки разработана собственная модель данных тайловой карты. Драйвер выполняет преобразование модели входных данных к модели данных библиотеки. Для интеграции таких подпрограмм в библиотеку разработан абстрактный интерфейс, определяющий набор функций, которые должна поддерживать программа драйвер. В статье алгоритмы реализации функций абстрактного интерфейса не рассматриваются.

Для формализованного описания программной архитектуры в статье будет использоваться язык моделирования UML.

Перечислим основные принципы, которые лежат в основе рассматриваемой технологии отображения тайловой карты на экране дисплея.

1. Использование программных изделий, разработанных в ФГУ ФНЦ НИИСИ РАН: операционная система реального времени семейства «Багет», графический сервер, библиотека интерфейсных компонентов и графическая библиотека.
2. Применение библиотеки функций «tilelib», инвариантной к модели данных тайловой карты.
3. Использование параллельных вычислений.

2. Программная архитектура

Разработанная программная архитектура позволяет создавать прикладные программы с графическим интерфейсом пользователя, предназначенные для отображения тайловых карт на экране дисплея (далее экране). Дисплей — это совокупность устройств, которыми управляет графический сервер. В состав дисплея входит (как минимум) один или несколько экранов, клавиатура и указывающее устройство (мышь); экраны нумеруются, начиная с нуля.

Программная архитектура включает в себя следующие компоненты: библиотека «tilelib», прикладная программа, драйвер, операционная система реального времени семейства «Багет», графический сервер, библиотека интерфейсных компонентов и графическая библиотека.

Функциональное назначение перечисленных выше программ следующее.

Операционная система реального времени семейства «Багет» предназначена для разработки программного обеспечения программно-аппаратных комплексов, работающих в режиме реального времени.

Библиотека «tilelib» реализует функции отображения тайловой карты на экране дисплея для языка программирования Си, а именно построение изображения тайловой карты, а также его сдвиг, поворот и масштабирование.

Драйвер — программа, реализующая функции абстрактного интерфейса, и обеспечивающая преобразование модели целевой тайловой карты к модели карты библиотеки «tilelib».

Прикладная программа — это программа, реализующая задачу пользователя.

Библиотека интерфейсных компонентов предназначена для программирования графического интерфейса пользователя. Библиотека содержит набор интерфейсных компонентов (или элементов), каждый из которых имеет определенное функциональное назначение. Примерами интерфейсных элементов служат кнопка, меню, окно диалога.

Графический сервер - это программа, которая управляет X-дисплеем.

Графическая библиотека, это библиотека, которая содержит функции, необходимые прикладным программам, выполняющимся на стороне клиента, для вывода графических примитивов распределённой графической системы X Window.

Модель программной архитектуры показана на рисунке 1 в виде диаграммы пакетов UML.

Пакет представляет собой механизм организации элементов в группы [7].

На диаграмме классы программ библиотека «tilelib», библиотека интерфейсных компонентов, графическая библиотека и драйвер сгруппированы в пакеты с именами «tilelib», «БИК», «ГБ» и «Драйвер» соответственно. Операционная система реального времени семейства «Багет», прикладная программа и графический сервер описываются классами с именами «ОС РВ», «Прикладная программа» и «ГС».

Класс – это описание множества объектов

с одинаковыми атрибутами, операциями, связями и семантикой [7]. Экземпляр класса будем называть объектом, а реализацию операции – методом.

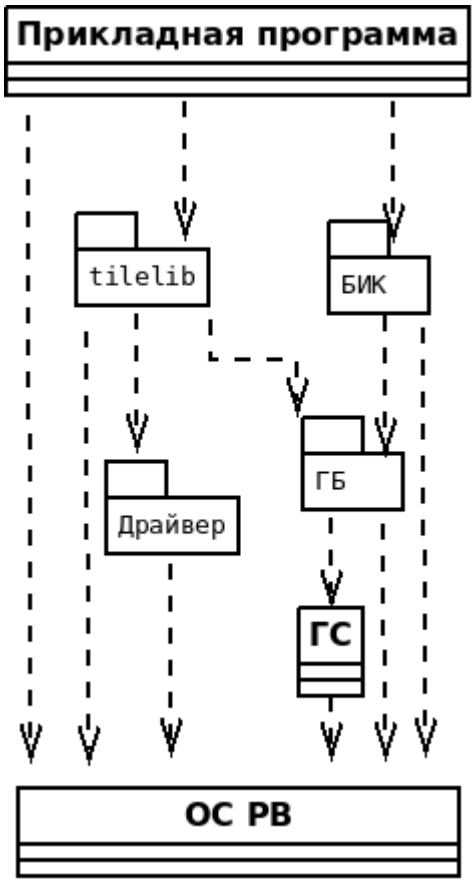


Рис. 1. Программная архитектура

Операционная система реального времени семейства «Багет» в программной модели рассматривается как системное и инструментальное программное обеспечение (ПО). В качестве системного ПО операционная система обеспечивает функционирование программ на ЭВМ "Багет-47-51". Как инструментальное ПО она является программной средой для разработки программных систем, в том числе реализует библиотеки функций. Вызовы методов библиотек объектами классов на диаграмме обозначены как связи зависимости между классом «ОС РВ» и пакетами «tilelib», «БИК», «ГБ», «Драйвер», а также классами «Прикладная программа» и «ГС».

В операциях классов пакета «tilelib» используется собственная модель данных тайловой карты. Для преобразования модели целевой тайловой карты к модели пакета «tilelib» применяются операции классов пакета «Драйвер». Указанное преобразование на диаграмме обозначается с помощью связи зависимости между пакетом «tilelib» и пакетом «Драйвер».

В прикладной программе вызываются методы объектов классов пакетов «tilelib» и «БИК» для отображения тайловой карты на экране и создания графического интерфейса пользователя, что показано с помощью связей зависимости между пакетом «Прикладная программа» и пакетами «tilelib» и «БИК».

В операциях классов пакета «tilelib» для вывода графической информации на экран используются операции пакета «ГБ», на что указывает связь зависимости между соответствующими пакетами.

Объекты классов пакета «БИК» передают сообщения объектам класса «ГС» с помощью вызова методов объектов классов пакета «ГБ». На диаграмме указанная последовательность передачи сообщений показана с помощью связей зависимости между соответствующими пакетами и классом «ГС».

4. Вычисление координат пикселя в декартовой системе координат

На диаграмме классов (см. рис. 2) показана модель растрового изображения, которое будем называть растром.

Рис. 2. Абстрактная модель растра

Класс «Растр» описывает данные растрового изображения, и является контейнером для сущностей класса «Пиксель», что показано с помощью связи композитного агрегирования между классами.

Понятие контейнер будет применятся к классам, которые имеют связь агрегирования с другими классами, являющимися его составными частями. Этот термин будет использоваться так же и для объектов классов. Далее в тексте и на диаграммах для обозначения объекта перед именем класса объекта будет ставиться двоеточие, а само название выделяться

курсивом. В случае необходимости именовать объект перед двоеточием будет указываться имя объекта [8].

Связь композитного агрегирования является разновидностью связи агрегирования, она означает, что «объект-часть может принадлежать только единственному целому, и, кроме того, как правило, жизненный цикл частей совпадает с циклом целого» [9].

Класс «Пиксель» описывает данные пикселя.

Атрибут «Color» класса «Пиксель» определяет код цвета объекта «:*Пиксель*».

Логически элементы класса «Пиксель» в контейнере «:*Растр*» структурированы в виде таблицы и находятся на пересечении столбцов и строк, номера которых обозначаются атрибутами X и Y соответственно, далее координаты объектов класса «Пиксель» будут обозначаться в виде (X,Y).

Объект «:*Растр*» может являться контейнером для сущностей своего класса, то есть растр можно получить из нескольких растров или наоборот из растра выделить фрагмент. Это свойство обозначено замкнутой на класс «Растр» связью агрегирования.

Для вычисления координат объекта «:*Пиксель*» введем декартову систему координат (СК). В СК объект «:*Пиксель*» будем еще называть точкой. Началом СК является пиксель с координатами (0,0) в контейнере «:*Растр*». По оси абсцисс будут нумероваться столбцы матрицы растра, а по оси ординат - строки. Тогда положительная ось абсцисс будет направлена вправо, а ординат - вниз, в соответствии с общими правилами нумерации столбцов и строк в таблице. В качестве линейной единицы измерения длин возьмем один пиксель, которую кратко будем обозначать «px». Линии границы растра, которые формируются из точек пикселей первых и последних столбцов и строк таблицы, образуют прямоугольник на плоскости. Ширина прямоугольника растра будет измеряться вдоль оси абсцисс, а высота – вдоль оси ординат.

Основной задачей, которую приходится решать при построении изображения, это расчет координат точки объекта «:*Пиксель*» в разных СК растров, при этом предполагается, что прямоугольники растров пересекаются на некотором множестве точек. Расчет выполняется с помощью формул преобразования координат.

Для примера, рассмотрим СК объектов «*Тайл:Растр*», «*Карта:Растр*» и «*Окно:Растр*» (см. Рис. 3).

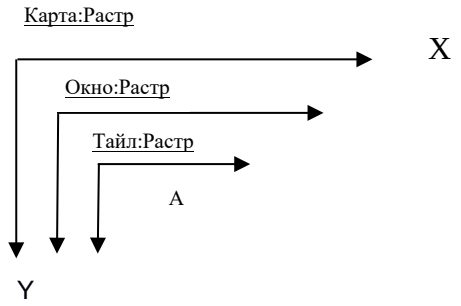


Рис. 3. СК объектов «*Тайл:Растр*», «*Карта:Растр*» и «*Окно:Растр*»

На рисунке показана ситуация, когда растр карты «собран» из растров ее тайлов и затем часть растра карты показана на экране в виде окна графической библиотеки.

В данном примере объект «*Окно:Растр*» является контейнером для объекта «*Карта:Растр*», который в свою очередь, является контейнером для объекта «*Тайл:Растр*». Все три объекта являются контейнерами для объекта «:*Пиксель*».

На рисунке СК объектов расположены таким образом, что имеют общую точку A. При этом точке A в контейнере «*Карта:Растр*» соответствует объект «*К:Пиксель*» с координатами точки (Xk,Yk), в контейнере «*Окно:Растр*» соответствует объект «*W:Пиксель*» с координатами точки (Xw,Yw) и в контейнере «*Тайл:Растр*» соответствует объект «*T:Пиксель*» с координатами точки (Xt,Yt).

Координаты точки A в разных СК рассчитываются с помощью формул преобразования координат. При этом надо знать смещение СК объектов «*Тайл:Растр*», «*Карта:Растр*» и «*Окно:Растр*» относительно друг друга, а так же координаты точки A в одной из СК. Примеры расчетов координат для некоторых задач технологии будут даны ниже.

5. Диаграмма классов пакета «tilelib»

Диаграмма классов пакета «tilelib» показана на рисунке 4.

Класс «Тайл» описывает данные тайла.

Атрибуты «T_WIDTH» и «T_HEIGHT» класса задают размер прямоугольника растра по ширине и высоте соответственно, по умолчанию значение этих атрибутов равно 256 px.

Атрибуты «X», «Y», «Z» задают адрес объекта «:*Тайл*» в контейнере «:*Карта*». В модели данных пакета «tilelib» логическая схема хранения тайлов реализована в виде таблицы T, которая состоит из строк Y и столбцов X на пересе-

чении которых находятся объекты «:*Тайл*». Каждый масштабный уровень Z контейнера «:*Карта*» имеет свою таблицу Tz.

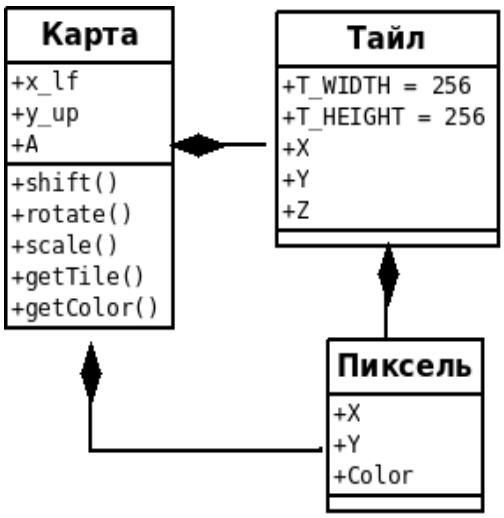


Рис. 4. Диаграмма классов пакета «tilelib»

Класс «Тайл» является контейнером для класса «Пиксель», на что указывает ассоциация композитного агрегирования между соответствующими классами.

Класс «Карта» описывает данные тайловой карты.

Он является контейнером для классов «Тайл» и «Пиксель», что показано с помощью ассоциации композитного агрегирования между соответствующими классами.

Класс «Карта» имеет следующие операции:

getTile() - операция создает объект «:*Тайл*»;

getColor() – операция возвращает значение атрибута «Color» объекта «:*Пиксель*»;

shift() – операция выполняют сдвиг СК растра объекта «:*Карта*» относительно СК объекта «:*Окно*» пакета «ГБ» (см. п. 7);

rotate() – операция выполняет поворот осей СК растра объекта «:*Карта*» относительно СК объекта «:*Окно*».

scale() - операция выполняет масштабирование карты.

Размер прямоугольника растра объекта «:*Карта*» по ширине и высоте можно посчитать по следующим формулам:

$$M\_WIDTH = Cx \times T\_WIDTH$$
$$M\_HEIGHT = Cy \times T\_HEIGHT$$

Где M_WDTH и M_HEIGHT– размер прямоугольника растра карты в пикселях по ширине и высоте соответственно, Cx и Cy – это количество объектов «:*Тайл*» по столбцам и строкам в таблице Tz для уровня масштабирования Z. T_WIDTH и T_HEIGHT– размер сторон прямоугольника растра объекта «:*Тайл*» соответственно по ширине и высоте в пикселях.

Атрибут «A» класса «Карта» задает угол поворота осей СК объекта «:*Карта*» в СК объекта «:*Окно*». Атрибуты «x_lf» и «y_up» задают координаты начала координат СК объекта «:*Карта*» в СК объекта «:*Окно*».

Класс «Пиксель» определяет данные пикселя.

Атрибут «Color» класса «Пиксель» задает код цвета объекта «:*Пиксель*».

Логически элементы класса «Пиксель» в контейнере «:*Тайл*» структурированы в виде таблицы и находятся на пересечении столбцов и строк, номера которых обозначаются атрибутами X и Y соответственно.

6. Диаграмма классов пакета «БИК»

Диаграмма классов пакета «БИК» показана на рисунке 5. Модель классов создана с использованием метода обратного проектирования.

Метод обратного проектирования – это способ построения модели путем анализа существующей программной системы [7].

Рис. 5. Диаграмма классов пакета «БИК»

Пакет «БИК» состоит из класса «Интерфейсный элемент», объекты которого предназначены для построения графического интерфейса пользователя. Класс «Интерфейсный элемент» описывает совокупность структур данных и операций, реализованных по определенным правилам, которые в процессе работы создают окно и обеспечивают реагирование на события, связанные с этим окном.

Объект «:*Интерфейсный элемент*» может являться контейнером по отношению к объектам своего класса. Это свойство обозначено замкнутой на класс «Интерфейсный элемент» связью агрегирования. Примером интерфейсного элемента, выполняющего функции контейнера для других элементов, является элемент «Контейнер» (Form).

В контрольной задаче использовались следующие интерфейсные элементы: «Ярлык» (Label), «Кнопка» (Command), «Тумблер» (Toggle), «Окно текстового редактирования» (AsciiText), «Бокс» (Box), «Контейнер» (Form).

7. Диаграмма классов пакета «ГБ»

Диаграмма классов пакета «ГБ» показана на рисунке 6.

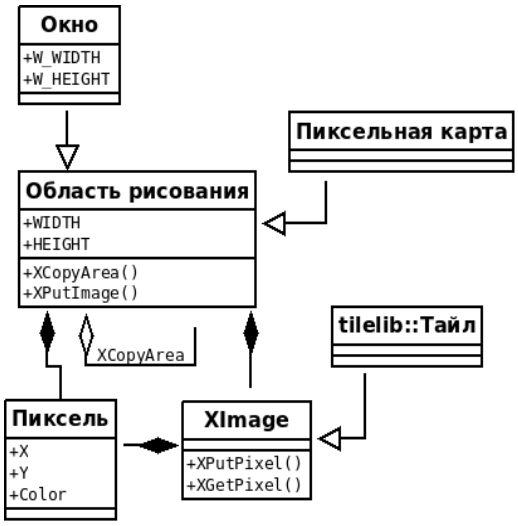


Рис. 6. Диаграмма классов пакета «ГБ»

На диаграмме указаны только те классы и операции пакета, которые участвуют в вычислении. Модель классов получена методом обратного проектирования.

Класс «Область рисования» описывает данные растра в формате, соответствующему типу «Drawable» графической библиотеки. Этот класс является контейнером для классов «XImage» и «Пиксель» на диаграмме эта связь показана с помощью ассоциаций композитного агрегирования между соответствующими классами. От этого класса наследуют свойства классы «Окно» и «Пиксельная карта» на диаграмме данные зависимости показаны с помощью связей обобщения.

Класс «Область рисования» имеет атрибуты «WIDTH» и «HEIGHT», которые задают размер прямоугольника растра соответственно по ширине и высоте в пикселях.

Объект «:*Область рисования*» может быть контейнером для сущностей своего класса. Это свойство обозначено замкнутой на класс «Область рисования» связью агрегирования с именем «XCopyArea».

Класс «Область рисования» имеет следующие операции:

XCopyArea() – эта операция выводит указанный прямоугольник изображения окна или пиксельной карты в указанную позицию окна или пиксельной карты.

XPutImage() - операция помещает объект «:*XImage*» в контейнер «:*Область рисования*».

Контейнер «:*Область рисования*» можно заполнить пикселями либо с использованием операции рисования точки XDrawPoint()(на диаграмме она не показана), либо с помощью операции XPutImage(). В последнем случае, сначала создаются объекты «:*XImage*» и заполняются пикселями с использованием операции XPutPixel(), а затем эти объекты помещаются в контейнер «:*Область рисования*» операцией XPutImage().

В технологии применяется способ заполнения контейнера «:*Область рисования*» с помощью операций XPutPixel() и XPutImage(), так как у него выше производительность. При этом используется несколько объектов «:*XImage*», чтобы иметь возможность обрабатывать их параллельно на нескольких процессорах.

Классы «Окно» и «Пиксельная карта» наследуют свойства класса «Область рисования». У этих классов имеется новое свойство данных – видимость пикселей на экране. Объекты «:*Пиксель*» контейнера «:*Окно*» отображаются на экране объектом «:*ГС*», а контейнера «:*Пиксельная карта*» - нет.

Класс «Окно» имеет атрибуты «W_WIDTH» и «W_HEIGHT», которые задают размер прямоугольника растра соответственно по ширине и высоте в пикселях.

Класс «Пиксель» описывает данные пикселя.

Он имеет атрибуты X и Y, которые задают адрес объекта «:*Пиксель*» в контейнерах «:*XImage*» и «:*Область рисования*» в формате столбец и строка.

Атрибут «Color» задает значение цвета пикселя.

Класс «XImage» описывает данные растра в формате «XImage» графической библиотеки.

Класс является контейнером для класса «Пиксель», на что указывает соответствующая ассоциация композитного агрегирования.

Класс «XImage» имеет следующие операции:

XPutPixel() – операция добавляет в контейнер «:XImage» объект «:*Пиксель*» и устанавливает значение его атрибута «Color».

XGetPixel() - операция возвращает значение атрибута «Color» объекта «:*Пиксель*».

Класс «tilelib::Тайл» пакета «tilelib» наследует свойства класса «XImage», что показано с помощью связей обобщения между классами.

8. Диаграмма классов пакета «Драйвер»

Диаграмма классов пакета «Драйвер» показана на рисунке 7.

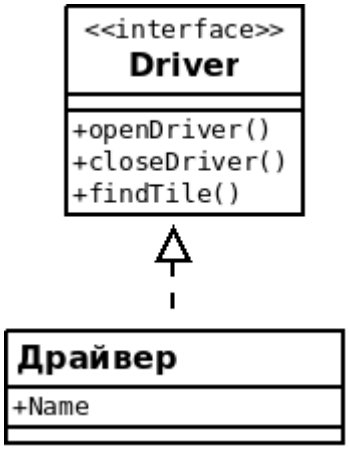


Рис. 7. Диаграмма классов пакета «Драйвер»

Класс «Драйвер» описывает объекты, которые выполняют преобразование модели данных целевой тайловой карты, к модели данных пакета «tilelib» и реализуют абстрактный интерфейс «Driver», на что указывает связь реализации между классом и интерфейсом.

В этой статье алгоритмы реализации объектом «*:Драйвер*» операций интерфейса «Driver» рассматриваться не будут.

Атрибут «Name» класса «Драйвер» задает имя объекта «*:Драйвер*», которое используется для его идентификации в операции openDriver().

Класс «Driver» со стереотипом <<interface>> описывает набор операций, которые должны быть реализованы объектом «*:Драйвер*». Это следующие операции:

openDriver() - операция конструктор, которая создает объект «*:Драйвер*»;

closeDriver() - операция деструктор, которая удаляет объект «*:Драйвер*».

findTile() – операция создает объект «*:Тайл*» с координатами (X,Y) с форматом, соответствующему модели данных карты пакета «tilelib».

9. Операция getTile() класса «Карта»

Операция getTile() класса «Карта» для объекта «*W:Пиксель*» с координатами (Xw,Yw) в контейнере «*:Окно*» создает объект «*M:Тайл*» с координатами (X,Y) в контейнере «*:Карта*». Если объект «*M:Тайл*» создать невозможно, то возвращается неопределенное значение, которое далее будем обозначать символом UNDEF.

Алгоритм выполнения операции getTile() следующий.

Найдем координаты (Хм,Ум) объекта «*M: Пиксель*» в контейнере «*:Карта*» через координаты (Xw,Yw) объекта «*W: Пиксель*» контейнера «*:Окно*».

Для этого надо рассмотреть две системы координат: систему координат объекта «*:Карта*» и СК объекта «*:Окно*». Начало координат СК объекта «*:Карта*» в СК объекта «*:Окно*» задается атрибутами «x_lf» и «y_up» класса «Карта».

Вычислим координаты (Хм,Ум) по следующим формулам:

$$Xм = Xw - x\_lf$$
$$Yм = Yw - y\_up$$

После этого вычислим координаты (X,Y) объекта «*M:Тайл*» в контейнере «*:Карта*» по следующим формулам:

$$X = MIN(Xм \div T\_WIDTH)$$
$$Y = MIN(Yм \div T\_HEIGHT)$$

Где функция MIN округляет результат в меньшую сторону, T_WIDTH и T_HEIGHT - это размер прямоугольника растра в пикселях объекта «*M:Тайл*» по ширине и высоте соответственно.

Зная координаты (X,Y) создадим объект «*M:Тайл*» с помощью метода findTile() объекта «*:Драйвер*».

10. Операция getColor() класса «Карта»

Операция getColor() класса «Карта» для объекта «*W:Пиксель*» с координатами (Xw,Yw) в контейнере «*:Окно*» возвращает значение атрибута «Color» объекта «*T:Пиксель*» с координатами (Xt,Yt) в контейнере «*M:Тайл*». Если объект «*T:Пиксель*» создать невозможно, то возвращается значение UNDEF.

Алгоритм выполнения операции getColor() следующий.

Для объекта «*W:Пиксель*» с координатами (Xw,Yw) создадим объект «*M:Тайл*» с помощью метода getTile() объекта «*:Карта*». Допустим, объект «*M:Тайл*» будет иметь координаты (X,Y) в контейнере «*:Карта*».

Далее определим координаты (Xt,Yt) объекта «*T:Пиксель*» в контейнере «*M:Тайл*».

Для этого сначала вычислим координаты (Xb,Yb) начала координат СК объекта «*M:Тайл*» в СК объекта «*:Карта*» по следующим формулам:

$$Xb = X \times T\_WIDTH$$
$$Yb = Y \times T\_HEIGHT$$

Где T_WIDTH и T_HEIGHT - это размер прямоугольника растра в пикселях объекта «*M:Тайл*» по ширине и высоте соответственно.

После этого найдем координаты (Xt,Yt) объекта «*T:Пиксель*» в контейнере «*M:Тайл*» по следующим формулам:

$$Xt = Xw - Xb - x\_lf$$
$$Yt = Yw - Yb - y\_up$$

Где x_lf , y_up это атрибуты объекта «*:Карта*», которые задают начало координат СК объекта «*:Карта*» в СК объекта «*:Окно*».

С помощью метода XGetPixel() объекта

«*M:Тайл*» получим значение атрибута «Color» объекта «*T:Пиксель*».

11. Операция «scale» класса «Карта»

Операция scale() создает объекты «*M:Тайл*» для определенного уровня масштабирования Z и размещает их в контейнере «*V:Окно*».

Алгоритм выполнения операции scale() следующий.

Прямоугольную область растра объекта «*V:Окно*» логически разделим на прямоугольные области P размером 256 px по ширине и 256 px по высоте.

Определим количество прямоугольников P по ширине Cx и высоте Cy по следующим формулам:

$$Cx = MAX(W\_WIDTH \div 256) + 1$$
$$Cy = MAX(W\_HEIGHT \div 256) + 1$$

Где функция MAX – это функция, которая округляет результат в большую сторону. W_WIDTH и W_HEIGHT - это размер прямоугольника растра в пикселях объекта «*V:Окно*» по ширине и высоте соответственно.

Будем рассматривать прямоугольник P как ячейку таблицы T с координатами X и Y, где X – это номер столбца таблицы, 0 <= X < Cx, а Y – это номер строки таблицы, 0 <= Y < Cy.

Далее создадим для каждого прямоугольника P объект «*M:Тайл*» контейнера «:*Карта*» и добавим его в контейнер «*V:Окно*» с координатами точки левого верхнего угла прямоугольника растра (x_draw, y_draw) в СК объекта «*V:Окно*» с помощью метода XPutImage().

Создадим объект «*M:Тайл*» следующим образом. Вычислим значение координат (Xw,Yw) в СК объекта «*V:Окно*» точки верхнего левого угла прямоугольника P по следующим формулам:

$$Xw = X \times 256$$
$$Yw = Y \times 256$$

Затем с помощью метода getTile() объекта «:*Карта*» создадим для объект «*W:Пиксель*» с координатами (Xw,Yw) объект «*M:Тайл*» контейнера «:*Карта*».

Чтобы добавить объект «*M:Тайл*» в контейнер «*V:Окно*» с помощью метода XPutImage(), найдем координаты (x_draw, y_draw) начала координат СК объекта «*M:Тайл*» в СК объекта «*V:Окно*» по следующим формулам:

$$x\_draw = Xb + x\_lf$$
$$y\_draw = Yb + y\_up$$

Где (Xb, Yb) – это координаты начала СК объекта «*M:Тайл*» в СК объекта «:*Карта*» (см. п. 9), x_lf , y_up это атрибуты объекта «:*Карта*», которые задают начало координат СК объекта «:*Карта*» в СК объекта «*V:Окно*».

Добавим объекты «*M:Тайл*» с координатами точки левого верхнего угла прямоугольника растра (x_draw, y_draw) в контейнер «*V:Окно*» с помощью метода XPutImage().

12. Операция «rotate» класса «Карта»

Операция rotate() класса «Карта» выполняет поворот осей СК объекта «:*Карта*» на заданный угол A относительно точки «*C: Пиксель*» с координатами (Xc,Yc) в СК объекта «*V:Окно*», а затем заполняет объектами «*W: Пиксель*» контейнер «*V:Окно*». Угол поворота A задается атрибутом «A» класса «Карта».

До выполнения поворота осей СК объекта «:*Карта*» каждому объекту «*T:Пиксель*» с координатами (Xt,Yt) в СК объекта «*M:Тайл*» контейнера «:*Карта*» соответствует объект «*W:Пиксель*» с координатами (Xw,Yw) в СК контейнера «*V:Окно*». После выполнения поворота осей СК объекта «:*Карта*» объект «*W:Пиксель*» будет уже иметь другие координаты (Xv,Yv) в СК объекта «*V:Окно*», которые можно найти по следующим формулам:

$$Xv = Xc + (Xw - Xc) \times \cos A - (Yw - Yc) \times \sin A$$
$$Yv = Yc + (Yw - Yc) \times \cos A + (Xw - Xc) \times \sin A$$

В предлагаемом алгоритме вычисление атрибута «Color» объектов «*W: Пиксель*» выполняется сегментами, а именно объект «*V:Окно*» логически делится на прямоугольные области размером 256 px по ширине и 256 px по высоте и для каждой области создается объект «*W:Тайл*». Каждый объект «*W:Тайл*» можно рассматривать как ячейку в таблице M, которая адресуется столбцом X и строкой Y.

Алгоритм выполнения операции rotate() следующий.

Предположим, что поворот осей СК объекта «:*Карта*» в СК объекта «*V:Окно*» уже выполнен.

Создадим объекты «*W:Тайл*» и заполним их объектами «*K:Пиксель*» с координатами (Xk,Yk).

Для начала определим значение атрибута «Color» каждого объекта «*K:Пиксель*».

Рассмотрим системы координат объектов «*W:Тайл*» и «*V:Окно*» и для объекта «*K:Пиксель*» с координатами (Xk,Yk) вычислим координаты (Xv,Yv) объекта «*W:Пиксель*» с помощью следующих формул:

$$Xv = Xk + Sx$$
$$Yv = Yk + Sy$$

Где Sx и Sy – это смещение СК объекта «*W:Тайл*» относительно СК объекта «*V:Окно*».

Вычислим смещения Sx и Sy по следующим формулам:

$$Sx = X \times 256$$
$$Sy = Y \times 256$$

После этого найдем координаты (Xw,Yw) объекта «*W:Пиксель*», которые он имел до поворота осей СК объекта «:*Карта*», по следующим формулам:

$$Xw = Xc + (Xv - Xc) \times \cos A + (Yv - Yc) \times \sin A$$

$$Yw = Yc + (Yv - Yc) \times \cos A - (Xv - Xc) \times \sin A$$

Далее определим значение атрибута «Color» с помощью метода getColor() для объекта «*W: Пиксель*» с координатами (Xw,Yw). Это значение так же будет значением атрибута «Color» объекта «*К:Пиксель*» с координатами (Xк,Yк) контейнера «*W:Тайл*».

После этого, с помощью метода XPutPixel() инициализируем значение атрибута «Color» объекта «*К:Пиксель*» с координатами (Xк,Yк) и поместим объект в контейнер «*W:Тайл*».

После загрузки объектов «*К:Пиксель*» в контейнеры «*W:Тайл*», поместим объекты «*W:Тайл*» в контейнер «*V:Окно*» с помощью метода XPutImage().

13 Операция «shift» класса «Карта»

Операция shift() выполняет логический сдвиг СК объекта «:*Карта*» относительно СК объекта «*V:Окно*». В предлагаемом методе такой логический сдвиг достигается за счет использования двух объектов «*V:Окно*» и «*N:Окно*», у которых размеры прямоугольников растра по ширине и высоте одинаковые. Объект «*V:Окно*» создается в результате выполнения операций rotate() или scale(). Объект «*N:Окно*» создается операцией shift() и существует до следующего вызова операций rotate() или scale().

Операция shift() вызывается в цикле, который завершается при выполнении операций rotate() или scale(). При каждой итерации операции shift() СК «*N:Окно*» смещаются относительно СК «*V:Окно*», при этом положение СК объекта «:*Карта*» относительно СК «*V:Окно*» не меняется. Координаты (x_lf, y_up) начала СК объекта «:*Карта*» в СК объекта «*V:Окно*» вычисляются в потоках управления операций rotate() и scale() на основе выходных данных операции shift().

При выполнении каждой итерации операции shift() СК объекта «*N:Окно*» сдвигается на Sx px по оси X или на Sy px по оси Y относительно СК объекта «*V:Окно*». Значения Sx и Sy являются постоянными величинами. Суммарная величина сдвига СК объекта «*N:Окно*» по осям X и Y в СК объекта «*V:Окно*» задается параметрами Xs и Ys соответственно и вычисляется по следующим формулам:

$$Xs = Sx \times count$$

$$Ys = Sy \times count$$

Где $count$ – это количество итераций операции shift().

После выполнения операций rotate() или scale() параметр count = 1, а точка начала СК «*N:Окно*» имеет координаты (0,0) в СК объекта «*V:Окно*».

В алгоритме операции shift() используется метод двойной буферизации для повышения скорости инициализации объекта «*N:Окно*».

Алгоритм выполнения операции shift() следующий.

Создадим объект «:*Пиксельная карта*».

Если count = 1 скопируем данные растра объекта «*V:Окно*» в объект «*N:Окно*» с помощью операции XCopyArea().

Если выполняется условие сдвига Sx < W_WIDTH и Sy < W_HEIGHT, где W_WIDTH и W_HEIGHT – это размеры прямоугольника растра объекта «*N:Окно*» по ширине и высоте соответственно, то с помощью метода XCopyArea() объекта «*N:Окно*» создадим объект «*A:Область рисования*» с размерами по ширине (W_WIDTH – Sx) px и высоте (W_HEIGHT – Sy) px и добавим его в контейнер «:*Пиксельная карта*» с координатами верхнего левого угла (Xa,Ya).

Объект «*A:Область рисования*» характеризуется тем, что у его объектов «*A:Пиксель*» известно значение атрибута «Color», который инициализируется в результате выполнения операций rotate() или scale(), если count = 1, или операции shift(), если count > 1.

Координаты (Xa,Ya) рассчитываются с учетом значений Sx и Sy. Пример программного кода операции shift() пакета «tilelib», где вычисляются координаты верхнего левого угла (Xa,Ya), показан ниже.

```
switch(direction){
        case UP:
XCopyArea( xcDisplay, pix_map.map1, pix_map.map2, pix_map.g_context, 0, step, width, height, 0, 0 );
            break;
        case DOWN:
XCopyArea( xcDisplay, pix_map.map1, pix_map.map2, pix_map.g_context, 0, 0, width, height, 0, step );
            break;
        case LEFT:
XCopyArea( xcDisplay, pix_map.map1, pix_map.map2, pix_map.g_context, step, 0, width, height, 0, 0 );
            break;
        case RIGHT:
XCopyArea( xcDisplay, pix_map.map1, pix_map.map2,pix_map.g_context, 0, 0, width,
```

```
height, step, 0 );
                break;
            default:
                break;
        }
```

В программном коде используется управляющая конструкция «переключатель», в которой в зависимости от значения переменной direction, определяющего направление сдвига, вызывается функция XCopyArea() с разными значениями аргументов, а именно изменяются значения пятого, шестого, девятого и десятого аргументов. Переменная step задает шаг сдвига или по оси X или по оси Y в зависимости от того, какое направление сдвига задано. Назначение аргументов функции XCopyArea() следующее. Первый аргумент является указателем на структуру, содержащую информацию о графическом сервере и экранах. Четвертый аргумент содержит указатель на структуру, которую называют графический контекст, значение атрибутов которой влияет на отображение графических объектов. Пятый и шестой аргументы задают координаты точки верхнего левого угла прямоугольника растра, который надо скопировать из области рисования, заданной вторым аргументом в область рисования, заданную третьим аргументом. Седьмой и восьмой аргумент задают размер прямоугольной области растра для копирования. Девятый и десятый аргументы задают координаты точки для верхнего левого угла копируемого растра в целевой области рисования, заданной третьим аргументом.

После добавления объекта «*A:Область рисования*» в контейнер «:*Пиксельная карта*», в прямоугольнике растра объекта «:*Пиксельная карта*» образуется прямоугольная область размером Sx px по ширине и Sy px по высоте, которая содержит объекты «*U:Пиксель*», для которых значение атрибута «Color» неизвестно.

Чтобы инициализировать атрибут «Color» объектов «*U:Пиксель*», создадим объект «*B:XImage*» с размером прямоугольника растра Sx px по ширине и Sy px по высоте и определим значение атрибута «Color» для всех объектов «*B:Пиксель*» объекта «*B:XImage*» после чего добавим его в контейнер «:*Пиксельная карта*» с координатами верхнего левого угла (Xb,Yb).

Для этого рассмотрим системы координат объектов «*B:XImage*», «*N:Окно*» и «*V:Окно*». Для объекта «*B:Пиксель*» с координатами (Xv,Yv) в СК объекта «*B:XImage*» найдем координаты (Xw, Yw) объекта «*W:Пиксель*» в контейнере «*V:Окно*» по следующим формулам:

$$Xw = Xv + Xs + Xb$$
$$Yw = Yv + Ys + Yb$$

Где (Xv, Yv) - координаты точки объекта «*B:Пиксель*» в СК объекта «*B:XImage*», (Xb, Yb) – координаты начала СК объекта «*B:XImage*» в СК «*N:Окно*», (Xs, Ys) – координаты начала СК объекта «*N:Окно*» в СК «*V:Окно*».

Теперь определим значение атрибута «Color» объекта «*W:Пиксель*» следующим образом. Если значение атрибута «A» объекта «:*Карта*» отлично от нуля и не кратно 2π, то значение атрибута «Color» определим по правилам операции rotate(), иначе значение атрибута «Color» получим с помощью операции getColor(). Это значение также будет значением атрибута «Color» объекта «*B:Пиксель*».

После того, как значение атрибута «Color» определено для всех объектов «*B:Пиксель*» контейнера «*B:XImage*», объект «*B:XImage*» добавим в контейнер «:*Пиксельная карта*» с помощью метода XPutImage() с координатами верхнего левого угла (Xb,Yb).

Затем объект «:*Пиксельная карта*» поместим в контейнер «*N:Окно*» с координатами верхнего левого угла (0,0) с помощью метода XCopyArea().

14. Поток управления прикладной программы

Диаграмма деятельности прикладной программы показана на рисунке 8.

В прикладной программе создаются объекты «:*Драйвер*», «:*Карта*» и «:*Интерфейсный элемент*», а также реализуется обработка событий, поступающих от интерфейсных элементов в ответ на действия оператора дисплея.

Деятельность «Создать интерфейс» создает группу объектов «:*Интерфейсный элемент*», которые генерируют программные события в ответ на действия с дисплеем оператора программы.

Деятельность «openDriver()» создает объект «:*Драйвер*».

Деятельность «Создать карту» создает объект «:*Карта*».

Далее следует цикл обработки событий генерируемых объектами «:*Интерфейсный элемент*». Одно событие резервируется для выхода из цикла, и когда оно наступает, происходит завершение потока управления прикладной программы.

Деятельность «closeDriver()» удаляет объект «:*Драйвер*».

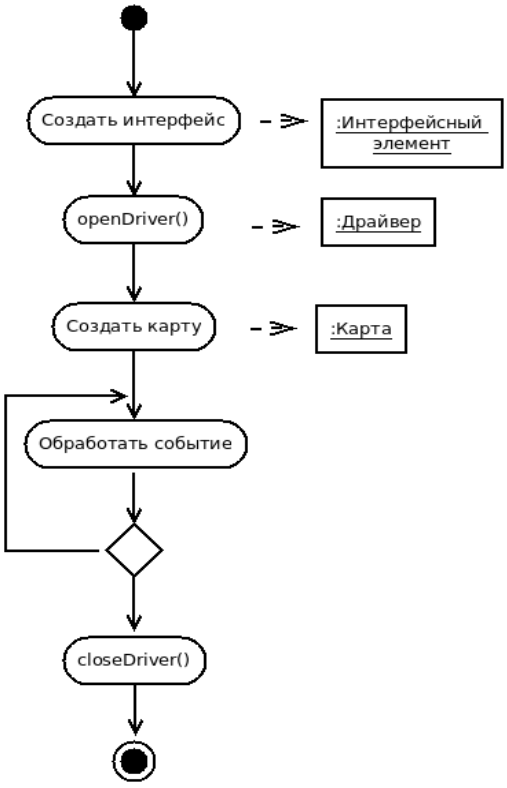


Рис. 8. Диаграмма деятельности прикладной программы

15. Заключение

В данной статье была рассмотрена технология отображения тайловой карты на экране дисплея с помощью программных средств для операционной системы реального времени семейства «Багет».

Была описана архитектура программной системы с использованием средств языка моделирования UML. В частности, были рассмотрены модели классов пакетов «ГБ», «tilelib» «БИК» и «Драйвер», описывающие статическое поведение системы, а также модель потока управления класса «Прикладная программа» в виде диаграммы деятельности и алгоритмы операций shift(), rotate(), scale(), getTile() и getColor() класса «Карта», описывающие динамическое поведение системы. Алгоритмы операций shift(), rotate(), scale() были разработаны таким образом, чтобы можно было использовать параллельные вычисления и повторно использовать информацию, и тем самым уменьшить время построения изображения тайловой карты. Модели классов пакетов «ГБ» и «БИК» построены методом обратного проектирования. Представленные в данной статье результаты были проверены с помощью контрольной задачи на ЭВМ "Багет-47-51". Проведенные измерения показали, что временные параметры визуализации геопространственных данных соответствуют заданным.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН (проведение фундаментальных научных исследований 47 ГП) по теме № 0065-2019-0002 «Исследование и реализация программной платформы для перспективных многоядерных процессоров» (рег. № АААА-А19-119012290074-2)

Description of the Technology for Constructing the Image of a Tile Map Using Software for the Real-Time Operating System

P.V. Egorov

Abstract. Tile map image construction using Baget family software tools.

Keywords: raster electronic map, tile map, tile, real-time.

Литература

1. ГОСТ 28441-99 Картография цифровая. Термины и определения.
2. Пиксель. https://ru.wikipedia.org/wiki/Пиксель (Дата обращения 17.11.2020).
3. TMS (Tile Map Service) http://nextgis.github.io/webgis_course/3/tms.html (Дата обращения 17.11.2020).
4. Тайловый картографический сервис. http://www.geocentre-consulting.ru/products/index?section=105 (Дата обращения 17.11.2020).
5. Теоретические сведения. https://tech.yandex.ru/maps/doc/jsapi/2.1/theory/index-docpage/ (Дата обращения 17.11.2020).
6. Tiled web map. https://en.wikipedia.org/w/index.php?title=Tiled_web_map&oldid=956139437

(Дата обращения 17.11.2020).

7. Гради Буч, Джеймс Рамбо, Ивар Якобсон «Язык UML. Руководство пользователя». Москва «Издательство ДМК Пресс» 2006 г.

8. П.В.Егоров Метод импорта пространственных данных формата "шейп-файл" в систему БГИСРВ. «Труды НИИСИ» Т. 9 № 1.

9. М. Фаулер, К. Скотт «UML в кратком изложении». Москва «Мир» 1999 г.

Определение коллизий аппроксимирующих параллелепипедов и цилиндров

Е.В. Страшнов[1], М.А. Торгашев[2], А.В. Мальцев[3]

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

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, mtorg@mail.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, avmaltcev@mail.ru

Аннотация. В работе рассматривается задача определения коллизий аппроксимирующих параллелепипедов (боксов) и цилиндров, окружающих геометрию виртуальных объектов. Для решения этой задачи предлагается подход, основанный на применении теоремы о разделяющей оси и анализе различных случаев контактного взаимодействия бокса и цилиндра. На основе такого подхода разработан алгоритм, позволяющий определить, как сам факт пересечения геометрий, так и необходимую контактную информацию о пересечении. Для поиска контактных точек задействованы быстрые тесты, основанные на методах отсечения грани бокса цилиндром, отрезка цилиндра боксом и вычислении ближайших точек между отрезками. Апробация предлагаемых в статье решений проводилась в программном комплексе виртуального окружения, созданном в ФГУ ФНЦ НИИСИ РАН, и показала их адекватность и эффективность.

Ключевые слова: определение коллизий, аппроксимирующий параллелепипед (бокс), аппроксимирующий цилиндр, контактное многообразие, теорема о разлеляющей оси, отсечение, система виртуального окружения

1. Введение

В системах виртуального окружения и имитационно-тренажерных комплексах в процессе компьютерного моделирования виртуальные объекты могут сталкиваться и пересекаться друг с другом. Это приводит к задаче определения коллизий (пересечений) этих объектов [1], [2]. Одним из подходов для решения этой задачи является использование аппроксимирующих контейнеров стандартной формы [3] (сфер, параллелепипедов, цилиндров, капсул и т.д.), которые окружают геометрию объектов. Тогда задача сводится к разработке методов и алгоритмов определения коллизий для каждой возможной пары таких контейнеров. В настоящей работе рассматривается задача определения пересечений аппроксимирующих параллелепипедов (боксов) с аппроксимирующими цилиндрами.

Здесь мы рассматриваем *апостериорные* алгоритмы определения коллизий виртуальных объектов (то есть те, которые обрабатывают коллизию по факту пересечения объектов). Такие алгоритмы условно подразделяются на две фазы: широкая и узкая. На стадии широкой фазы выполняется грубая проверка на пересечение объектов, выходом которой является список пар, которые потенциально могут пересекаться друг с другом. Примером служит алгоритм «Sweep and prune» сортировки AABB [4] (*англ.* Axis Aligned Bounding Boxes – боксов, выравненных по осям мировой системы координат). Полученный на широкой фазе список пар контейнеров затем используется на узкой стадии определения коллизий, в которой проверяется не только сам факт пересечения, но и вычисляется контактная информация о пересечении (точки контакта, глубины проникновения и нормали расталкивания). На основе этой информации вычисляется реакция на столкновение (разрешение коллизий [5]), включая расчет сил и моментов, действующих на объекты.

В данной статье мы рассматриваем алгоритмы узкой стадии определения коллизий виртуальных объектов (в частности, коллизия бокса и цилиндра), которой в последние годы было посвящено множество работ. Существующие решения включают алгоритмы, предназначенные для определения коллизий выпуклых объектов. Один из подходов относится к семейству итерационных алгоритмов GJK-EPA [6]-[8], в которых объекты задаются в виде выпуклой оболочки множества точек [9]. В общем случае для сложных геометрий эти алгоритмы имеют плохую сходимость и обладают слабой численной устойчивостью. Альтернативой являются алгоритмы, основанные на теореме о разделяющей оси SAT [10], [11] (*англ.* Separating Axis Theorem). Одно из решений заключается в аппроксимации цилиндра правильной призмой. Это приводит к огромному числу возможных разделяющих осей, для сокращения которых применяется сферическое отображение бокса и цилиндра [12]. Точность вычислений при таком подходе сильно зависит от аппроксимации цилиндра, а количе-

ство разделяющих осей все равно остается большим. В настоящей работе для решения этой проблемы предлагается алгоритм, в котором разделяющие оси вычисляются в зависимости от возможных случаев контакта бокса и цилиндра. Такой подход обладает приемлемой точностью, а количество возможных разделяющих осей не превышает 39. В свою очередь для вычисления контактных точек был разработан алгоритм, в котором применяется отсечение ближайших элементов бокса и цилиндра. Предложенные в статье алгоритмы были реализованы в программном комплексе виртуального окружения и апробированы на примерах взаимодействия объектов, имеющих форму бокса и цилиндра.

2. Постановка задачи

Рассмотрим аппроксимирующий бокс, полудлины которого равны l_i, $i=\overline{1,3}$. Пусть локальная система координат (ЛСК) бокса расположена в его центре, имеет начало в точке O_B и оси, заданные единичными ортами $\mathbf{e}_i$, как показано на рис. 1.

Также рассмотрим аппроксимирующий цилиндр радиусом *r* и высотой *h*, центр которого расположен в точке O_C, а ось имеет направление, заданное единичным вектором *v*. Основаниями цилиндра являются круги с центрами C_1 и C_2. Все значения координат точек и векторов задаются в мировой системе координат (МСК).

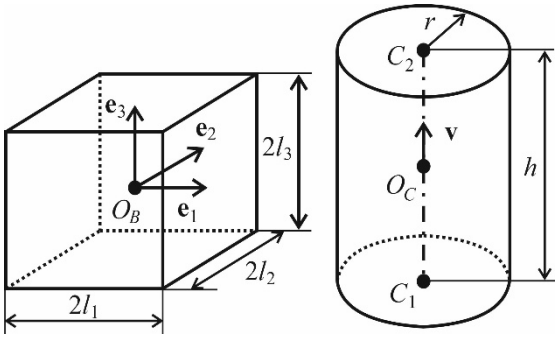


Рис. 1. Бокс и цилиндр

Задача определения коллизий виртуальных объектов заключается в проверке пересечения объектов и вычислении информации о пересечении в виде *контактного многообразия* (*англ.* contact manifold). Контактное многообразие задается набором параметров $C_P=\{P,\mathbf{n},d\}$, где *P* – точка контакта, **n** – единичный вектор нормали в точке *P*, *d* – глубина проникновения. При рассмотрении пересечения аппроксимирующего бокса и цилиндра будем искать точки P_i, которые находятся на поверхности бокса, их глубины проникновения d_i, $i=\overline{1,M}$, где *M* – количество точек, а также нормаль **n**, направленная от бокса к цилиндру. Далее подробно опишем предлагаемые методы и алгоритмы для решения этой задачи.

3. Алгоритм определения коллизий бокса и цилиндра

Для определения коллизий аппроксимирующего бокса и цилиндра воспользуемся теоремой о разделяющей оси (см. [10], [11]). Идея этой теоремы заключается в том, что два выпуклых объекта не пересекаются, если найдется плоскость, которая их разделяет. Тогда первый объект находится по одну сторону от плоскости, а второй – по другую. Нормаль к этой плоскости соответствует направлению разделяющей оси. Важным следствием этой теоремы является то, что глубина проникновения двух геометрий равна минимальной длине отрезка пересечения геометрий на некоторую ось. Таким образом, теорема о разделяющей оси позволяет не только определить пересечение двух объектов, но также вычислить нормаль и глубину проникновения.

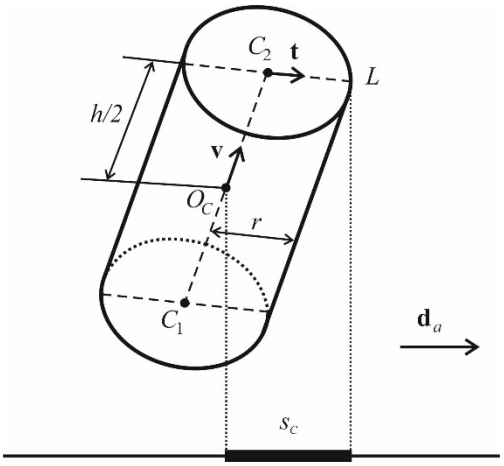


Рис. 2. Проекция цилиндра на ось

Спроецируем аппроксимирующий бокс и цилиндр на разделяющую ось, которая задается единичным вектором $\mathbf{d}_a$. Для проецирования бокса рассмотрим вектор $e_B=l_1\mathbf{e}_1+l_2\mathbf{e}_2+l_3\mathbf{e}_3$. Тогда в силу симметрии бокса величина

$$s_B=\left|\mathbf{e}_B\cdot\mathbf{d}_a\right| \qquad (1)$$

равна половине отрезка проекции бокса на ось.

Для проецирования цилиндра на ось рассмотрим точку *L* (см. рис. 2). В силу симметрии цилиндра величина $s_C=\left|\mathbf{O}_C\mathbf{L}\cdot\mathbf{d}_a\right|$ (модуль проекции вектора $\mathbf{O}_C\mathbf{L}$ на ось $\mathbf{d}_a$) равна половине отрезка проекции цилиндра на ось. Вектор $\mathbf{O}_C\mathbf{L}$ вычисляется как

$$\mathbf{O}_C\mathbf{L} = \mathbf{O}_C\mathbf{C}_2 + \mathbf{C}_2\mathbf{L} = \frac{h}{2}\mathbf{v} + r\mathbf{t}, \qquad (2)$$

где $\mathbf{t}$ – вектор, лежащий в плоскости основания цилиндра.

Можно показать, что для вектора $\mathbf{t}$ будет справедливо равенство

$$\mathbf{t}\cdot\mathbf{d}_a = \sqrt{1-(\mathbf{v}\cdot\mathbf{d}_a)^2}\,. \qquad (3)$$

С учетом (2) и (3) половина проекции цилиндра на ось вычисляется как

$$s_C = \left|\mathbf{O}_C\mathbf{L}\cdot\mathbf{d}_a\right| = \left|\frac{h}{2}\mathbf{v}\cdot\mathbf{d}_a + r\sqrt{1-(\mathbf{v}\cdot\mathbf{d}_a)^2}\right|. \qquad (4)$$

Рассмотрим вектор $\mathbf{T} = \mathbf{O}_B\mathbf{O}_C$, соединяющий центры бокса и цилиндра. Если существует ось $\mathbf{d}_a$, для которой будет выполнено неравенство

$$\left|\mathbf{T}\cdot\mathbf{d}_a\right| > s_B + s_C,$$

то бокс и цилиндр не пересекаются.

Разделяющие оси выбираются в зависимости от возможных случаев контактного взаимодействия бокса и цилиндра. На рис. 3 приведены возможные случаи контакта: а) – основание цилиндра с гранью бокса, б) – боковая поверхность цилиндра с гранью бокса, в) – основание цилиндра с ребром бокса, г) – боковая поверхность цилиндра с ребром бокса, д) – боковая поверхность цилиндра с вершиной бокса, е) – окружность основания цилиндра с ребром бокса.

При контакте цилиндра с гранью бокса (рис. 3а,б) разделяющей осью является нормаль к грани бокса, т.е. $\mathbf{d}_a = \mathbf{e}_i$, $i = \overline{1,3}$.

Для случая контакта основания цилиндра с ребром бокса (рис. 3в) или вершиной бокса разделяющая ось параллельна оси цилиндра $\mathbf{d}_a = \mathbf{v}$.

В свою очередь, при контакте боковой поверхности цилиндра с ребром бокса (рис. 3г) разделяющая ось вычисляется через векторное произведение ребра бокса и вектора оси цилиндра: $\mathbf{d}_a = \mathbf{e}_i \times \mathbf{v}$.

При контакте боковой поверхности цилиндра с вершиной бокса A_j, $j = \overline{1,8}$, как показано на рис. 3д, разделяющая ось перпендикулярна оси цилиндра и плоскости, которая проходит через вершину A_j. Эта плоскость определяется с помощью векторов $\mathbf{v}$ и $\mathbf{A}_j\mathbf{O}_C \times \mathbf{v}$, а разделяющая ось является нормалью к этой плоскости, т.е. $\mathbf{d}_a = \mathbf{v}\times(\mathbf{A}_j\mathbf{O}_C \times \mathbf{v})$.

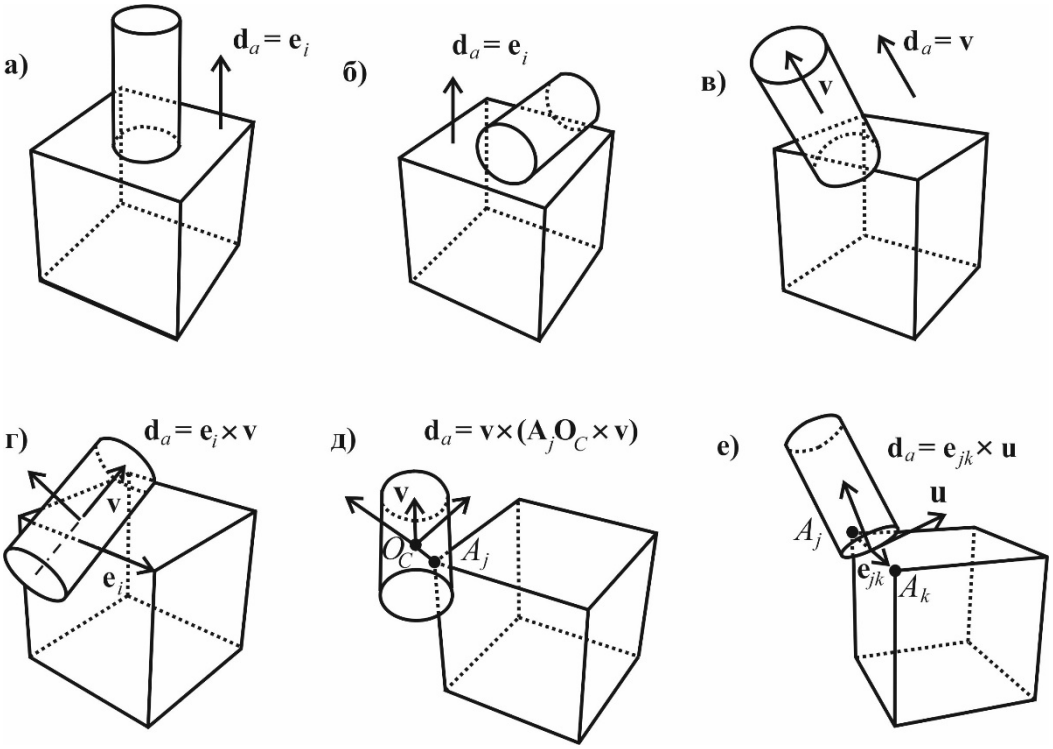


Рис. 3. Возможные случаи контакта бокса и цилиндра

Наконец последним случаем является контакт окружности основания цилиндра с ребром бокса (рис. 3е). Для поиска разделяющей оси рассмотрим ребро $\mathbf{A}_j\mathbf{A}_k$ бокса и основание цилиндра (например, нижнее с центром C_1). В этом случае разделяющей осью будет вектор

$\mathbf{d}_a = \mathbf{e}_{jk} \times \mathbf{u}$, где $\mathbf{e}_{jk} = \mathbf{A}_j\mathbf{A}_k / \left\|\mathbf{A}_j\mathbf{A}_k\right\|$ – единичный вектор ребра, а $\mathbf{u}$ – касательный вектор окружности, проходящий через точку касания окружности с ребром бокса. Разделяющая ось существует в том случае, если ось цилиндра не перпендикулярна ребру, т.е. $\mathbf{e}_{jk} \cdot \mathbf{v} \neq 0$.

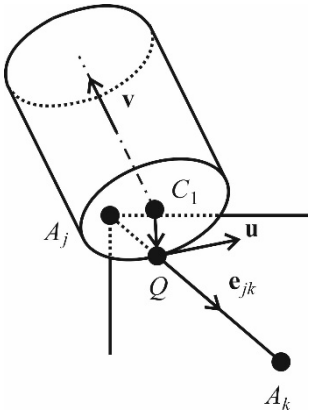


Рис. 4. Контакт окружности основания цилиндра с ребром бокса

Покажем, как вычисляется касательный вектор $\mathbf{u}$ (см. рис. 4). Для этого найдем точку Q пересечения отрезка $\mathbf{A}_j\mathbf{A}_k$ с плоскостью основания цилиндра. Из параметрического представления отрезка получим, что

$$Q = A_j + s\mathbf{e}_{jk}, \qquad (5)$$

где s – параметр отрезка.

Вектор $\mathbf{C}_1\mathbf{Q}$ находится в плоскости основания цилиндра, поэтому перпендикулярен вектору $\mathbf{v}$. Это приводит к условию

$$\mathbf{C}_1\mathbf{Q} \cdot \mathbf{v} = 0. \qquad (6)$$

Подставляя (5) в (6), получим

$$s = \frac{\mathbf{A}_j\mathbf{C}_1 \cdot \mathbf{v}}{\mathbf{e}_{jk} \cdot \mathbf{v}}.$$

Тогда из (5) следует, что

$$Q = A_j + \mathbf{e}_{jk}\frac{\mathbf{A}_j\mathbf{C}_1 \cdot \mathbf{v}}{\mathbf{e}_{jk} \cdot \mathbf{v}}.$$

В итоге получим, что касательный вектор $\mathbf{u}$ перпендикулярен плоскости, образуемой векторами $\mathbf{C}_1\mathbf{Q}$ и $\mathbf{v}$, и вычисляется как

$$\mathbf{u} = \mathbf{C}_1\mathbf{Q} \times \mathbf{v}.$$

Таким образом, для случая контакта основания цилиндра с ребром бокса имеем 24 возможных разделяющих осей, полученных для верхнего и нижнего основания цилиндра, а также для каждого ребра бокса.

В таблице 1 приведены различные типы контактного взаимодействия бокса и цилиндра. Для каждого из этих типов указаны векторы разделяющих осей и их количество N.

Таблица 1. Разделяющие оси

Тип контакта	$\mathbf{d}_a$	N
Цилиндр – грань бокса	$\mathbf{e}_i$, $i = 1,2,3$	3
Основание цилиндра – ребро бокса	$\mathbf{v}$	1
Боковая поверхность цилиндра – ребро бокса	$\mathbf{e}_i \times \mathbf{v}$, $i = 1,2,3$	3
Боковая поверхность цилиндра – вершина бокса	$\mathbf{v} \times (\mathbf{A}_j\mathbf{O}_C \times \mathbf{v})$, $j = \overline{1,8}$	8
Окружность основания цилиндра – ребро бокса	$\mathbf{e}_{jk} \times \mathbf{u}$, $j,k \in \{1,2,3\}$	24

Алгоритм определения коллизий цилиндра с боксом на основе теоремы о разделяющей оси будет следующим:

1. Инициализация $d = d_{\max}$.

2. Цикл k = 1,..,N по всем разделяющим осям $\mathbf{d}_a^k$ (N = 39):

2.1. Вычисляем вектор $\mathbf{T} = \mathbf{O}_B\mathbf{O}_C$, а также проекции бокса s_B и цилиндра s_C на разделяющую ось по формулам (1) и (4).

2.2. Если $\left|\mathbf{T} \cdot \mathbf{d}_a^k\right| > s_B + s_C$, то досрочный выход. Бокс и цилиндр не пересекаются.

2.3. Вычисляем $\Delta d = s_B + s_C - \left|\mathbf{T} \cdot \mathbf{d}_a^k\right|$.

2.4. Если $\Delta d < d$, то $d = \Delta d$ и $\mathbf{n} = \mathbf{d}_a^k$.

Конец алгоритма.

4. Вычисление контактных точек

В случае, если бокс и цилиндр пересекаются, требуется вычислить контактные точки P_i и их глубины проникновения d_i, $i = \overline{1,M}$. Предлагаемое решение для вычисления контактных точек включает случаи, когда нормаль параллельна и не параллельна оси цилиндра, а также случай, когда нормаль является векторным произведением ребра бокса и вектора оси цилиндра. Рассмотрим эти случаи более подробно.

4.1. Нормаль параллельна оси цилиндра

В этом случае контактные точки вычисляются путем отсечения грани бокса цилиндром. Так как нормаль направлена от бокса к цилиндру, то выбирается та грань бокса, которая наиболее удалена в направлении нормали. Цилиндр аппроксимируется правильной много-

угольной призмой, плоскости которой участвуют в отсечении.

Покажем, как выполняется отсечение грани (многоугольника) плоскостью. Плоскость отсечения задается уравнением

$$s(P) = k_P - P \cdot \mathbf{n}_P, \qquad (7)$$

где P – точка, принадлежащая плоскости; $\mathbf{n}_P$ – нормаль к плоскости; k_P – постоянная плоскости.

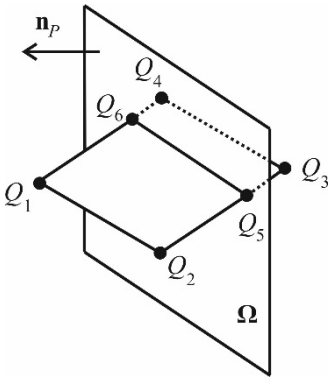


Рис. 5. Отсечение прямоугольника плоскостью

Рассмотрим очередное отсечение. Входом является многоугольник с вершинами Q_k, который был получен при выполнении предыдущих отсечений, где $k = \overline{1,K}$. Для каждого ребра $\mathbf{Q}_i\mathbf{Q}_j$, $i, j \in \{1, K\}$ этого многоугольника проверяется расположение вершин относительно плоскости. Если из уравнения (7) выполнено $s(Q_i) \geq 0$ и $s(Q_j) \geq 0$, то это ребро находится перед плоскостью. При $s(Q_i) < 0$ и $s(Q_j) < 0$ ребро располагается позади плоскости. Если $s(Q_i) \geq 0$ и $s(Q_j) < 0$, то ребро пересекает плоскость. Точка пересечения добавляется в вершины нового многоугольника. Те вершины, которые находятся перед плоскостью сохраняются, а вершины, находящиеся позади плоскости, удаляются. Задача отсечения заключается в том, чтобы сформировать новый многоугольник, все вершины которого находятся перед плоскостью. На рис. 5 показан результат отсечения прямоугольника плоскостью Ω. Исходный многоугольник (прямоугольник) включает вершины Q_k, $k = \overline{1,4}$, а после отсечения получим многоугольник с вершинами Q_k, k =1,2,5,6.

Результат отсечения грани бокса восьмиугольной правильной призмой показан на рис. 6 на примере случая контакта основания цилиндра с гранью. На выходе отсечения получаем многоугольник, вершины которого P_i, $i = \overline{1,8}$ являются точками контакта.

Глубины проникновения точек контакта вычисляются как

$$d_i = \mathbf{P}_i\mathbf{P}_i' \cdot \mathbf{n},$$

где P_i' – проекция точек P_i на ближайшее основание цилиндра.

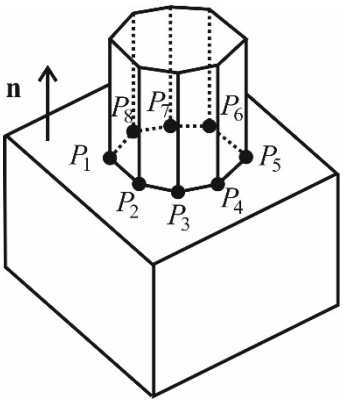


Рис. 6. Отсечение грани бокса призмой

4.2. Нормаль не параллельна оси цилиндра

Если нормаль произвольна относительно оси цилиндра, то для поиска контактных точек выполняем отсечение отрезка $\mathbf{Q}_1\mathbf{Q}_2$ боковой поверхности цилиндра боксом. Этот отрезок образуется путем проецирования оси цилиндра в направлении, противоположном нормали, как показано на рис. 7 для случая контакта боковой поверхности цилиндра с гранью бокса.

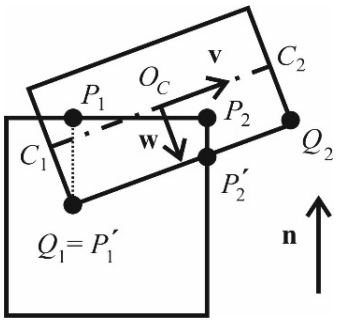


Рис. 7. Отсечение отрезка цилиндра боксом

Тогда получим, что

$$Q_i = C_i + \mathbf{w},$$

где $\mathbf{w} = r\dfrac{\mathbf{v}\cdot\mathbf{n} - \mathbf{n}}{\|\mathbf{v}\cdot\mathbf{n} - \mathbf{n}\|}$.

Отсечение отрезка $\mathbf{Q}_1\mathbf{Q}_2$ выполняется для каждой грани бокса и на выходе получаем отрезок $\mathbf{P}_1'\mathbf{P}_2'$. Точки контакта находим путем проецирования этого отрезка на грань бокса:

$$P_i = P_i' + d_i\mathbf{n}, \; i = 1,2,$$

где глубины проникновения точек контакта с

учетом (1) вычисляются следующим образом

$$d_i = s_B - \mathbf{O}_B\mathbf{P}_i' \cdot \mathbf{n}\,.$$

4.3. Контакт боковой поверхности цилиндра с ребром бокса

При контакте боковой поверхности цилиндра с ребром бокса нормалью является векторное произведение ребра бокса и вектора оси цилиндра. Пусть $\mathbf{A}_1\mathbf{A}_2$ – ребро бокса, наиболее удаленное в направлении нормали, а $\mathbf{Q}_1\mathbf{Q}_2$ – отрезок боковой поверхности цилиндра. Рассмотрим отрезок $\mathbf{PP}'$, который является общим перпендикуляром к $\mathbf{A}_1\mathbf{A}_2$ и $\mathbf{Q}_1\mathbf{Q}_2$ (см. рис. 8), где P – точка контакта. С примененим параметрического задания отрезков получим, что

$$P = A_1 + \alpha\mathbf{e}\,,\ P' = Q_1 + \beta\mathbf{v}\,, \qquad (8)$$

где $\mathbf{e}$ – единичный вектор направления ребра, α и β – неизвестные параметры.

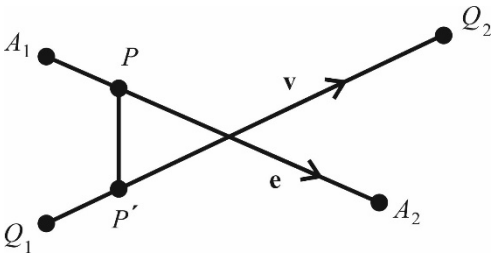


Рис. 8. Ближайшие точки двух отрезков

Отсюда верно, что

$$\mathbf{PP}' = \mathbf{A}_1\mathbf{Q}_1 + \beta\mathbf{v} - \alpha\mathbf{e}\,. \qquad (9)$$

Так как $\mathbf{PP}' \perp \mathbf{e}$ и $\mathbf{PP}' \perp \mathbf{v}$, то

$$\mathbf{PP}' \cdot \mathbf{e} = 0\,,\ \mathbf{PP}' \cdot \mathbf{v} = 0\,. \qquad (10)$$

Подставляя (9) в (10), получим систему линейных уравнений относительно α и β:

$$\alpha - (\mathbf{e}\cdot\mathbf{v})\beta = \mathbf{A}_1\mathbf{Q}_1 \cdot \mathbf{e}\,;$$

$$-(\mathbf{e}\cdot\mathbf{v})\alpha + \beta = -\mathbf{A}_1\mathbf{Q}_1 \cdot \mathbf{v}\,.$$

Если $\mathbf{e}$ и $\mathbf{v}$ не параллельны ($\mathbf{e}\cdot\mathbf{v} \neq \pm 1$), то, решая эту систему уравнений, находим

$$\alpha = \frac{1}{1-(\mathbf{e}\cdot\mathbf{v})^2}(\mathbf{A}_1\mathbf{Q}_1 \cdot \mathbf{e} - (\mathbf{e}\cdot\mathbf{v})\mathbf{A}_1\mathbf{Q}_1 \cdot \mathbf{v})\,.$$

Подставляя это выражение в (8), получим соотношение для вычисления точки контакта P.

5. Результаты моделирования

Предложенные в статье методы и алгоритмы определения коллизий аппроксимирующих боксов и цилиндров были реализованы в программном комплексе виртуального окружения, созданном в ФГУ ФНЦ НИИСИ РАН. В этом комплексе была проведена апробация на примерах различных случаев контактного взаимодействия бокса и цилиндра. Для этого было проведено сравнение предлагаемого в статье метода с подходом, основанном на аппроксимации цилиндра правильной призмой и применения сферического отображения цилиндра и бокса для нахождения потенциальных разделяющих осей [11].

Таблица 2. Сравнение результатов

Тип контакта	**Предл. метод**	**Призма n = 16**	**Призма n = 32**
I	d = 3.000	d = 3.000 (N = 26)	d = 3.000 (N = 50)
II	d = 3.000	d = 3.000 (N = 30)	d = 3.000 (N = 54)
III	d = 2.999	d = 2.999 (N = 52)	d = 2.999 (N = 108)
IV	d = 2.999	d = 2.999 (N = 52)	d = 2.999 (N = 108)
V	d = 2.999	d = 3.001 (N = 64)	d = 3.001 (N = 126)
VI	d = 2.999	d = 3.079 (N = 66)	d = 3.072 (N = 130)

Для апробации была создана виртуальная сцена, в которой бокс имеет полудлины $l_1 = l_2 = 40$ см, $l_3 = 20$ см, а цилиндр – радиус r = 30 см и высоту h = 80 см. В каждом случае контактного взаимодействия было задано пересечение бокса и цилиндра на глубину d = 3 см. На рис. 9 показан результат моделирования в системе виртуального окружения, где синим изображены точки контакта.

В таблице 2 приведены результаты сравнения для различных типов контакта: I – основания цилиндра с гранью бокса, II – боковой поверхности цилиндра с гранью бокса, III – основания цилиндра с ребром бокса, IV – боковой поверхности цилиндра с ребром бокса, V – боковой поверхности цилиндра с вершиной бокса, VI – окружности основания цилиндра с ребром бокса. Для аппроксимации цилиндра была рассмотрена правильная многоугольная призма с n = 16 и n = 32, где n – число вершин основания призмы. Из моделирования следует, что для случаев I - IV оба подхода дают идентичные результаты. При этом для случаев I и II с аппроксимацией цилиндра призмой с n = 16 требуется меньше потенциальных разделяющих осей (N = 26 и 30 против 39). Однако для случаев V и VI предлагаемый подход дает более точный результат и требуется меньше разделяющих осей (N = 39 против 64 и 66 для призмы с n = 16).

Предлагаемые в статье решения были реализованы для моделирования виртуальных колесных роботов. На рис. 10 показано взаимодействие колеса робота КРТ-100МП с объектом, имеющим форму бокса.

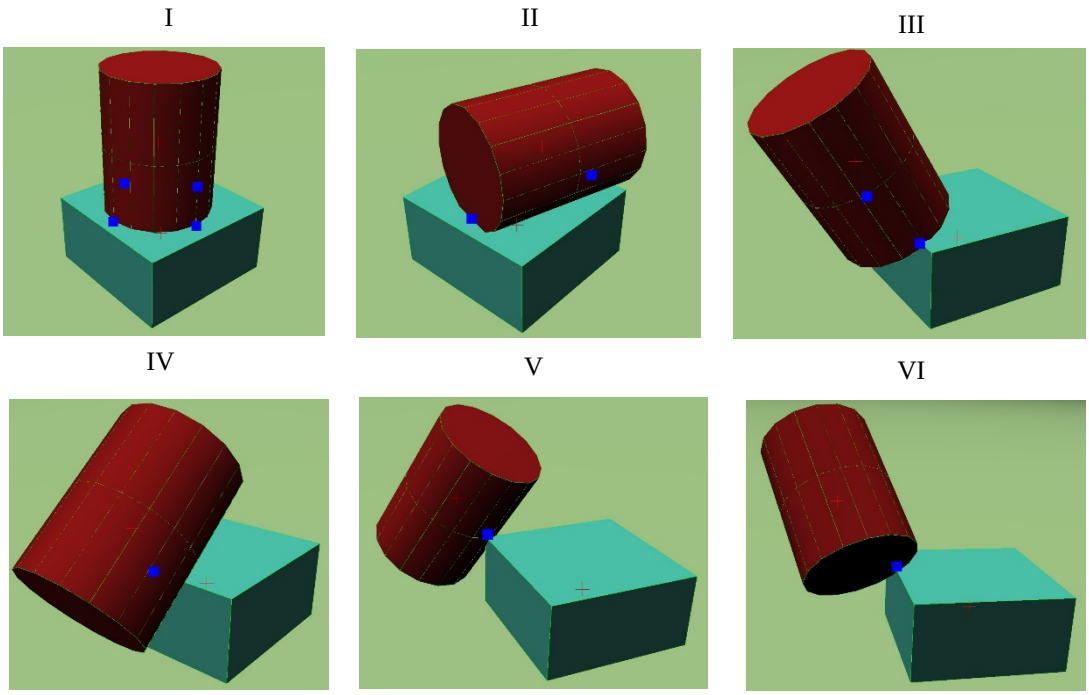


Рис. 9. Моделирование контакта бокса и цилиндра

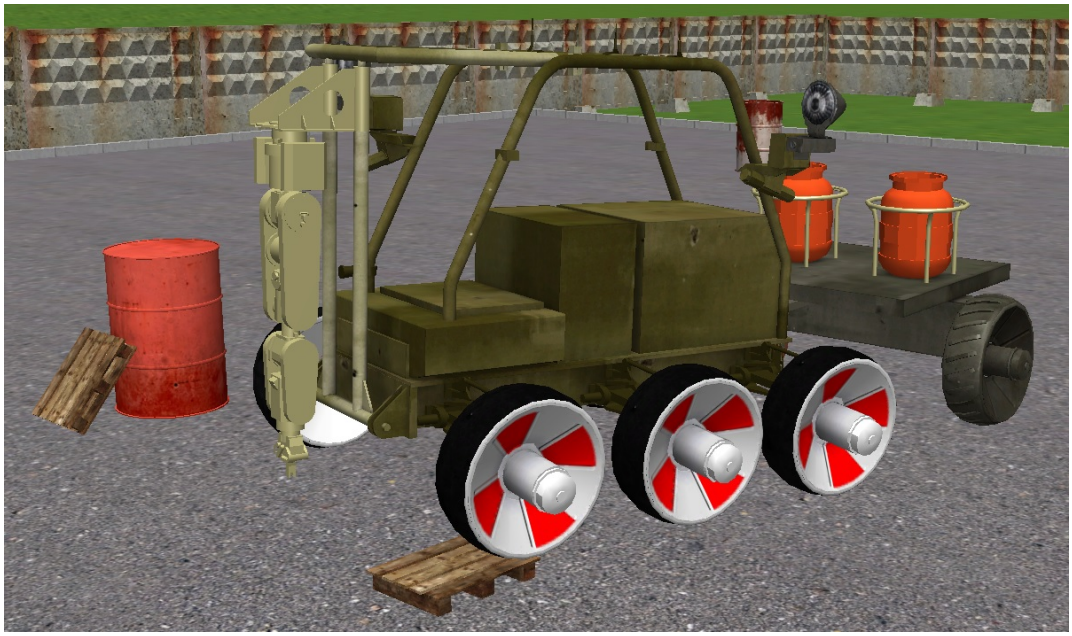


Рис. 10. Моделирование колесного робота КРТ-100МП

6. Заключение

В данной работе предложены методы и алгоритмы определения коллизий аппроксимирующих боксов и цилиндров, основанные на анализе возможных случаев контактного взаимодействия при выборе разделяющих осей. Апробация в системе виртуального окружения показала адекватность и эффективность предлагаемого решения. В дальнейшем планируется создание методов и алгоритмов определения коллизий аппроксимирующих цилиндров друг с другом.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-07-00387.

Collision Detection of Bounding Boxes and Cylinders

Evgeny Strashnov, Mikhail Torgashev, Andrey Maltsev

Abstract. The paper considers the problem of collision detection of bounding parallelepipeds (boxes) and cylinders surrounding the geometry of virtual objects. To solve this problem, an approach is proposed based on the application of the separating axis theorem and the analysis of various cases of contact interaction between the box and the cylinder. On the basis of this approach, an algorithm has been developed that makes it possible to detect both the fact of intersection of geometries and the necessary contact information about the intersection. To find for contact points, quick tests are used, based on the method for clipping the box face with a cylinder, cylinder segment with a box, and calculating the nearest points between the segments. The approbation of the solutions proposed in the paper was carried out in the virtual environment software complex, created at the SRISA RAS, and showed their adequacy and effectiveness.

Keywords: collision detection, bounding parallelepiped (box), bounding cylinder, contact manifold, separating axis theorem, clipping, virtual environment system

Литература

1. C. Ericson. Real-time collision detection. CRC Press, 2004.

2. G. van den Bergen. Collision detection in interactive 3D environments. Morgan Kaufmann Publishers, San Francisco, 2004.

3. Bounding volume. https://en.wikipedia.org/wiki/Bounding_volume (дата обращения: 26.10.2020).

4. J. Cohen, M. Lin, D. Manocha, and M. Ponagmi. I-collide: An interactive and exact collision detection system for large-scale environments. «Proc. of ACM Interactive 3D Graphics Conference», 1995, 189-196.

5. А.М. Трушин. Обработка коллизий виртуальных объектов с помощью метода последовательных импульсов. «Труды НИИСИ РАН», Т. 4 (2014), № 2, 95-105.

6. G. van den Bergen. A Fast and robust GJK Implementation for collision detection of convex objects, «Journal of Graphics Tools», vol. 4 (1999), no. 2, 7-25.

7. Y.-K. Choi, X. Li, W. Wang, S. Cameron. Collision detection of convex polyhedra based on duality transformation, Technical Report TR-2005-01, Department of Computer Science, The University of Hong-Kong, 2005.

8. А.В. Санников. Численное моделирование динамики систем твердых деформируемых и жестких тел. Диссертация на соискание ученой степени кандидата физ.-мат. наук, Москва, 2015.

9. D. Gregorius. Implementing quickhull. «Game Developers Conference (Valve Software)», San Francisco, 2014.

10. J. Huynh. Separating axis theorem for oriented bounding boxes, 2009. https://www.jkh.me/files/tutorials/Separating%20Axis%20Theorem%20for%20Oriented%20Bounding%20Boxes.pdf (дата обращения: 26.10.2020).

11. D. Gregorius. The separating axis test. «Game Developers Conference», 2013.

12. G. van den Bergen, and D. Gregorius. Game physics pearls. AK Peters/CRC Press, 2010.

Построение компилятора-интерпретатора для гибридной текстово-пиктограммной цифровой образовательной среды ПиктоМир-К

Д.Б. Аглямутдинова[1], М.В.Райко[2], А.Г. Леонов[3]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, dagliamutdinova@vip.niisi.ru;

[2]ФГУ ФНЦ НИИСИ РАН, г. Москва, Россия, mila.rayko@gmail.com;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россиядr.l@vip.niisi.ru

Аннотация. В современном постиндустриальном мире знакомство учеников с основами алгоритмизации должно происходить в раннем возрасте, еще до школы. Цифровые образовательные среды должны предоставить педагогу и ученику методически удобный механизм для раннего освоения алгоритмов, снабженный пиктографическим языком для составления программ дошкольниками, которые еще не умеют читать, или не любят читать. Однако, последующий переход от дошкольного пиктографического к школьному текстовому программированию оказывается для юных программистов непростым. Гибридная цифровая образовательная среда ПиктоМир-К сочетает в себе простоту привычной «сборки» программ из пиктограмм с текстовым представлением алгоритма, и, фактически, является мостом, позволяющим школьнику легко перейти пиктографических к учебным и производственным языкам программирования.

В статье рассматривается способ представления и выполнения программ, сконструированных в цифровой образовательной среде ПиктоМир-К, включая описание языка программирования с синтаксисом, семантикой и алфавитом. Также описывается структура данных синтаксического дерева программы и ее узлов, рассматривается алгоритм создания набора инструкций, соответствующего синтаксическому дереву, и приводится механизм выполнения этих инструкций с помощью виртуальной стековой машины, основанной на базе исполняющей машины среды ПиктоМир.

Ключевые слова: ПиктоМир, цифровая образовательная среда, компилятор, стековая машина, программирование

1. Введение

Проникновение в повседневную жизнь цифровых информационных технологий нарастает с каждым днем. Как реакция на эту тенденцию в мировой и отечественной системах образования нарастает интерес к раннему преподаванию программирования, как одного из важнейших разделов информатики [1]. В России это интерес разделяют законодательные ветви власти [2]. Все возрастающая потребность общества в специалистах в области информационных технологий ставит перед образованием задачи создания простых, пропедевтических курсов по программированию, способных поддерживать интерес у начинающих (дошкольников, школьников или студентов) и(или) студентов), ограждая новичков от концептуальных сложностей освоения профессиональных сред программирования и необходимости предварительного овладения навыками эффективного клавиатурного ввода и познаниями в английском языке, желательными для понимания синтаксиса англоязычных языков программирования [3].

Разработка цифровой образовательной среды ПиктоМир [4] началась в ФГУ ФНЦ НИИСИ РАН 10 лет назад, как среды для «первого знакомства с программированием» [5]. Эта среда и построенные на ее базе пропедевтические курсы оказались эффективными для обучаемых различных возрастов [6], [7], [8], [9] и [10]. В ПиктоМире ученик как бы собирает программу, составляя ее из кубиков-пиктограмм, каждый из которых обозначает действие или вопрос, или число (которое можно использовать в качестве задания числа повторений выполнения действия или группы действий). Действия отличаются по форме от вопросов, а те, в свою очередь от повторителей, что исключает «попадание» пиктограммы не на свое место. Таким образом программа ребенка, собранная им из пиктограмм всегда синтаксически корректна, что позволяет ученику сосредоточиться только на придумывании алгоритма. ПиктоМир позволяет педагогу постепенно знакомить детей с основными понятиями программирования, при этом в каком бы

возрасте ученик не начинал изучать азы алгоритмизации, ему все равно придется освоить (в том числе на практике) эти важнейшие понятия [11]. На практике оказывается, что эти базовые понятия программирования вполне доступны детям дошкольного и младшего школьного возраста, не умеющими или не слишком любящими читать и писать.

Гибридная, пиктограммно-текстовая цифровая образовательная среда ПиктоМир-К [12], обладающая некоторой частью функционала школьного алгоритмического (текстового) языка, используемого в учебнике основной школы [13], предлагает набор инструментов, достаточный не только для обучения новичков, но и являющимся связующим звеном в образовательном процессе между пиктограммной и текстовой стилями программирования, что позволяет существенно упростить переход от визуального представления алгоритма к более производственному, стандартному [1]. При этом процесс составления программы, так же, как и в ПиктоМире, позволяет всегда поддерживать синтаксически корректное состояние учебной программы, что является существенной поддержкой для ученика, позволяющей ему сконцентрироваться на содержательной задаче – придумывании и составлении алгоритма, не тратя драгоценного времени на исправлении синтаксических ошибок в программе [14].

Проблема, возникающая у многих новичков, начинающих изучение программирования с популярных промышленных языков, таких как C++, Java, Python, заключается в необходимости одновременного решения трех сложных задач: освоения среды разработки, изучение синтаксиса языка и понимания и (или) придумывания алгоритма для решения учебной задачи. По нашему опыту работы с новичками самого разного возраста и уровня подготовки, такой подход к преподаванию программирования новичкам является менее эффективным по сравнению с методикой преподаванием при помощи цифровых образовательных сред ПиктоМир и ПиктоМир-К. Последние сознательно ограничены в функционале по сравнению с полноценными производственными средами разработки и адаптированы под начинающего программиста.

Простота языка позволяет сконструировать достаточно простой компилятор для языка, что избавляет от необходимости использования тяжеловесных библиотек для организации процессов компиляции и выполнения программ.

2. Язык программирования Пикто-К

Программа на языке Пикто-К составляется, как и в ПиктоМире, с помощью пиктограмм, однако при этом сразу после вставки пиктограммы на ее место в программе, пиктограмма преобразуется в текст на школьном алгоритмическом языке. Ниже синтаксис языка описывается синтаксическими диаграммами Вирта [15] (см. рис.1).

Каждая программа в среде ПиктоМир-К фактически связана с конкретным исполнителем (большинство из них унаследованы из ЦОС ПиктоМир): Вертун, Двигун, Чертежник, Кузнечик, Черепаха, Водолей, Робот, Ползун, Зажигун и т.п. Исполнитель полностью определен списком предписаний (командами, методами), каждому методу соответствует своя уникальная пиктограмма.

Программа начинается с команды подключения используемого исполнителя, основной алгоритм не имеет имени и параметров, для вспомогательных алгоритмов предусмотрено пять предопределенных имен - пять первых букв алфавита. Вспомогательные алгоритмы, в отличие от основного, могут иметь параметры (например, исполнители Черепаха и Чертежник имеют методы с входными параметрами для указания длины линии или координат). С помощью предписаний-команд-методов обучающийся может перемещать исполнителя по полю, закрашивать клетки, рисовать линии и тому подобное в зависимости от типа исполнителя. На рисунке 1 представлена программа на языке Пикто-К.

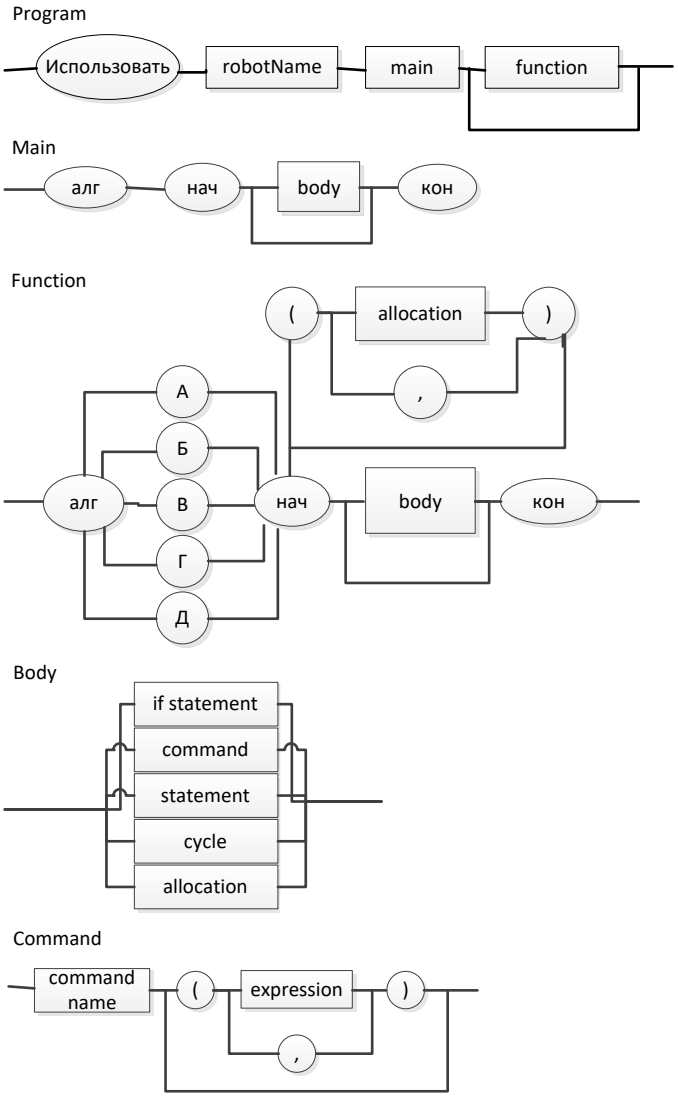


Рис. 1. Программа на языке Пикто-К

Язык Пикто-К включает в себя несколько видов циклов (на рисунке 2 приведено формальное описание цикла), условные конструкции, а также действия для работы с переменными-величинами: объявление переменных, присваивание им значений, простейшие арифметические действия. Язык ограничен двумя типами данных - логический и целый.

Условные конструкции на языке Пикто-К представлены двумя видами: «если -то» и «если-то-иначе». В качестве условий используются логические выражения, логические переменные и константы («да» и «нет»). Описание условных конструкций на языке Пикто-К приведено на рисунке 3.

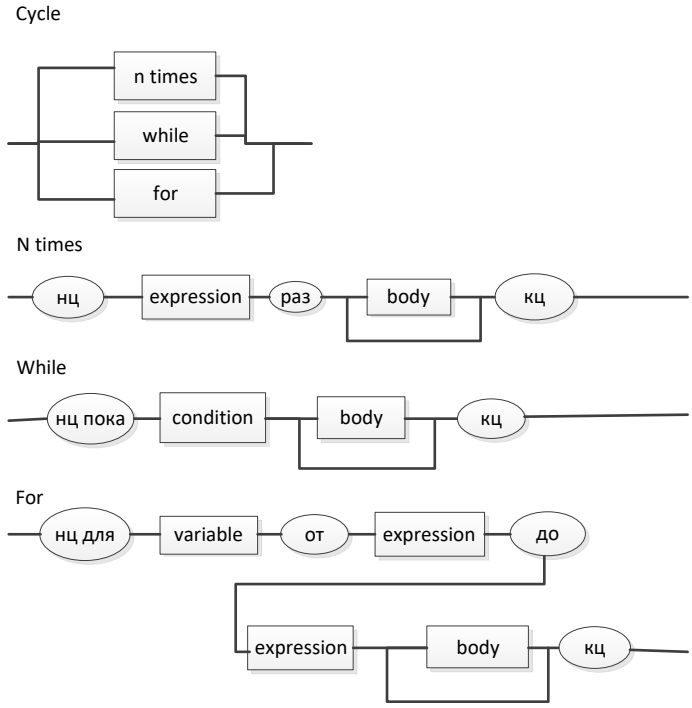


Рис. 2. Синтаксис конструкций циклов на языке Пикто-К

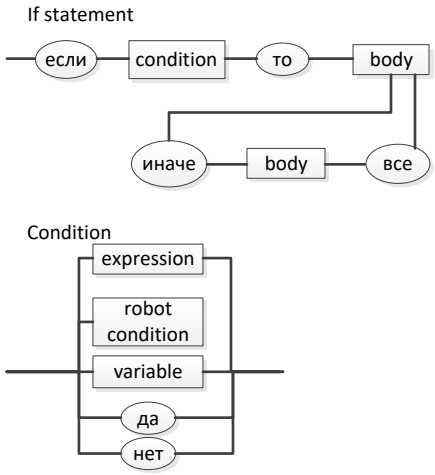


Рис. 3. Синтаксис условных конструкций на языке Пикто-К

Работа с величинами начинается с добавления пиктограммы объявления переменной в поле программы, при этом автоматически генерируется уникальное имя, обозначенное на рисунке 4 как «variable». После объявления переменной становятся доступны команды-пиктограммы инициализации переменной и ее значения. При этом допускается только корректная с точки зрения синтаксиса языка вставка.

Целой переменной можно присвоить целое значение, а логической только логическое. При удалении объявления переменной автоматически удаляются все ее вхождения в программу. Ниже описаны диаграммы оперирования с переменными, где number обозначает целые числа.

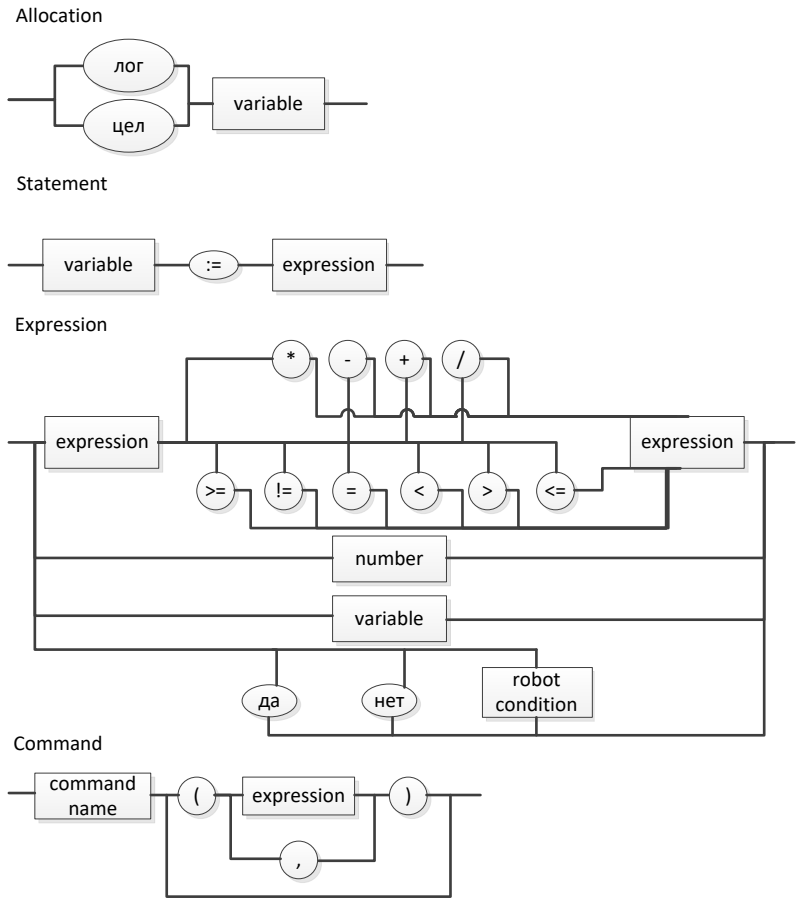


Рис. 4. Синтаксис оперирования переменными на языке Пикто-К

Грамматику алгоритмического языка Пикто-К можно описать в форме Бэкуса-Наура в следующем виде:

<program> ::= “использовать” <robot name> <line end>
<main function><line end>
{<algorithm function> <line end>}.

<main function> ::= “алг” <line end>
“нач” <line end>
{assignment <line end>}
“кон”.

<algorithm function> ::= “алг” <identifier> <line end>
“нач” <line end>
{<assignment> <line end>}
“кон”.

<assignment> ::= <statement>|<command>|<loop>|<if_statement>.

<loop> ::= <ntimes> | <while> | <for>.
<ntimes> ::= “нц” <expression> “раз” <line end>
{<assignment> <line end>}
“кц”.

<while>::=”нц пока” <condition> <line end>
{<assignment> <line end>}
“кц”.
<for> ::= “нц для> <identifier> “от” <expression>
“до” <expression> <line end>
{<assignment> <line end>}
“кц”.

<if statement> ::= “если” <condition> <line end>
“то” <line end>
{<assignment> <line end>}
[“иначе” <line end>>
{<assignment> <line end>}]
“все”.

<condition> ::= <expression> | <bool> | <identifier> | <condition name>.

<statement> ::= <identifier> "::=" <expression>.

<method> ::= <method name> [{<expression>}].

<expression>::= <expression> "+" <expression>|
<expression> "-" <expression>|
<expression> "*" <expression>|
<expression> "/" <expression>|
<expression> ">" <expression>|
<expression> "<" <expression>|
<expression> ":=" <expression>|
<expression> "!::=" <expression>|
< number> | <identifier>.

<alloc> ::= “цел” | “лог” <identifier>.

<identifier> ::= <alphabetic character> { <alphabetic character> | <digit> } .
<number >::= <int> | <bool>.
<int> ::= ["-"], <digit>, {<digit>}.
<bool> ::= "0" | "1".

где <alphabetic character> - это символы алфавита,
<digit> - это цифра,
<method name> - название метода робота,
<robot name> - имя исполнителя,
<condition name> - название условия текущего исполнителя,
<line end> - это конец строки.

3. Структура данных синтаксического дерева

В ЦОС ПиктоМир-К программа представляется синтаксическим деревом, в котором множество вершин представляет собой составные части программы. Каждая вершина этого дерева имеет ссылку на соответствующий ей элемент визуализируемой части программы. Ниже описаны типы вершин синтаксического дерева, определяющего программу.

На рисунке 5 изображен пример программы для исполнителя Робот в среде ПиктоМир-К. В алгоритме используется условный цикл “пока” с условием “снизу свободно” и команды “вниз” и “закрасить”. На рисунке 6 изображено синтаксическое дерево этой программы, узлы дерева

представлены окружностями, а стрелками обозначены дочерние элементы узла. В таблице 1 представлены типы вершин синтаксического дерева языка ПиктоМир-К.

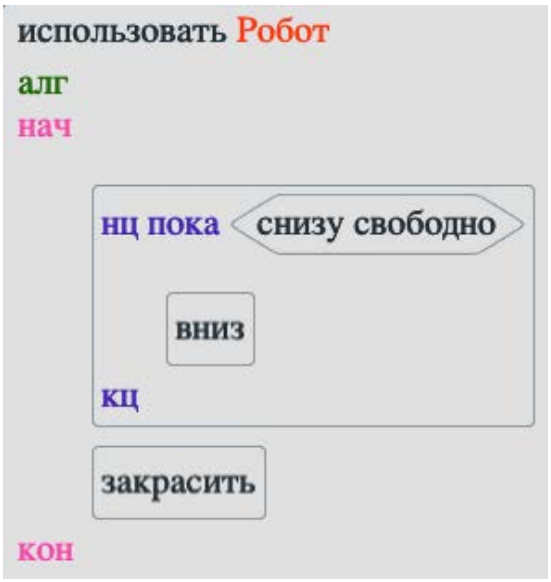


Рис.5. Программа в среде ПиктоМир-К

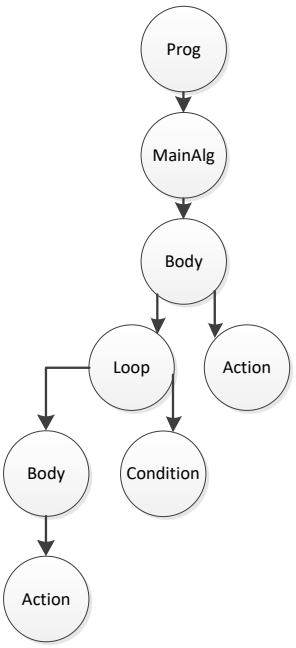


Рис. 6. Синтаксическое дерево программы

Таблица 1. Типы вершин синтаксического дерева языка ПиктоМир-К

Тип	Значение	Описание
Prog	-	Корень дерева, описывающий программу
MainAlg	-	Функция main
Func	А, Б, В, Г, Д	Функция
Body	“then_body”, “else_body”	Тело какой-либо конструкции
Statement	-	Операция присваивания
IfStatement	“if”, “if_else”	Условная конструкция
Loop		Цикл
Type	“bool”, “int”	Тип переменной
Identifier	“a1, a2, …”, “л1, л2, ..»	Идентификатор переменной
Number	0 - 999	Число
Bool	«да», «нет»	Логическое значение
Expression	“plus”, “minus”, “mult”, “devide”	Выражение
Condition	Команды вопросы	Условие

	конкретного робота.	
LogicExpression	“less”, “more”, “equal”, ”not_equal”	Логическое выражение
Action	Методы конкретного робота или название вызываемой функции.	Метод робота
Alloc	-	Аллокация новой переменной
Empty: 'empty'	-	Пустая вершина

4. Компиляция программы

На этапе компиляции осуществляется обход синтаксического дерева с генерацией соответствующих инструкций. В результате создается набор инструкций для виртуальной стековой машины (см. табл. 2). Эта виртуальная машина выполняет определенные действия в зависимости от типа инструкции и ее данных.

Таблица 2. Описание инструкций виртуальной машины

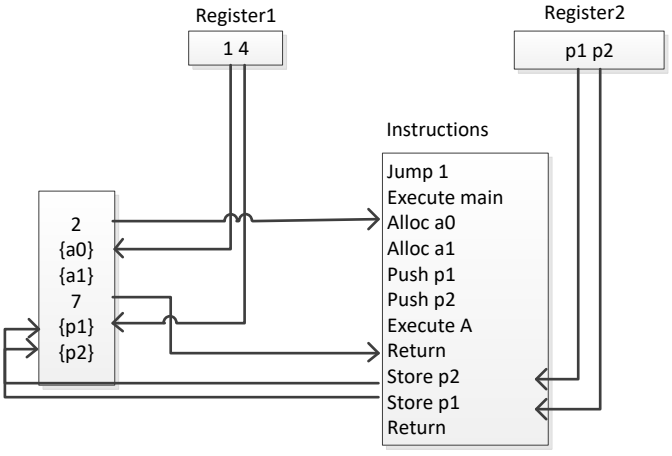
№	Тип	Описание	Аргументы
0	Execute	Выполняет метод робота либо подпрограмму. При выполнении подпрограммы с вершины стека достаются параметры, в стек заносится текущая позиция в списке инструкций, а в регистр вызовов функций заносится текущая позиция в стеке для определения области локальных данных этого метода.	isNative– является ли вызовом подпрограммы methodID – уникальный идентификатор метода робота paramCount– количество параметров, которые нужно взять из регистра.
1	Check Condition	Выполняет проверку условия для робота и кладет результат на вершину стека.	robot – ссылка на исполнителя condition – уникальный идентификатор условия для проверки роботом.
2	Start loop	Является индикатором начала цикла, задает начальное значение итератору цикла. Достает с вершины стека текущее значение итератора и конечное, если текущее значение > конечного, то переходит на конец блока цикла. В противном случае кладет на вершину стека начальное и конечное значение итератора.	blockEndLabel – ссылка на конец блока цикла repType – тип цикла.
3	End loop	Достает с вершины стека текущее и конечное значения итератора. Если итератор <= конечному значению, то выполняет переход в начало цикла, увеличивает итератор на единицу и кладет в стек текущее значение итератора и конечное.	jumpLabel – ссылка для перехода для выполнение следующей итерации цикла robot – ссылка на исполнителя.
4	Return	Выполняет выход из подпрограммы, если есть выходные параметры, то достает их из вершины стека и кладет в регистр параметров, далее получает из	paramQuantity – количество параметров, которые нужно взять из регистра.

		стека вызовов функций позицию, начиная с которой в нем расположены локальные данные этой функции. Удаляет все локальные данные и последнее значение из регистра вызовов. Если регистр вызовов пуст – программа завершилась.	
5	Jump_if	Переход по ссылке, если условие на вершине стека верно.	jumpLabel – ссылка для перехода.
6	Jump_nif	Переход по ссылке, если условие на вершине стека неверно.	jumpLabel – ссылка для перехода.
7	Jump	Безусловный переход по ссылке.	jumpLabel – ссылка для перехода.
8	Alloc	Выделяет место для переменной в памяти, если вызов осуществляется вне функции или кладет переменную в стек, если в регистре вызовов функции уже есть вызов.	varType – тип переменной varName – имя переменной value – значение переменной.
9	Fetch	Кладет на вершину стека значение переменной, ищет значение этой переменной в стеке в области видимости этой функции.	varName – имя переменной.
10	Push	Кладет в стек значение.	value – значение.
11	Store	Сохраняет значение с вершины стека или из регистра параметров в переменной.	varName – имя переменной fromReg – взять значение из регистра параметров или из стека
12	Pop	Удалить последнее значение из стека.	-
13	Calc	Выполняет операцию над последними двумя значениями с вершины стека.	operation – операция.

5. Устройство виртуальной исполняющей машины

Виртуальная исполняющая машина состоит из стека, регистров и памяти. Стек хранит локальные данные функций и операций. Регистр вызовов функций сохраняет в себе позицию в стеке, начиная с которой расположены локальные данные вызванной функции. Регистр параметров сохраняет входные и выходные параметры функций. Память – это ассоциативный массив глобальных переменных.

Например, для программы, представленной на рисунке 7, набор инструкций будет выглядеть так:



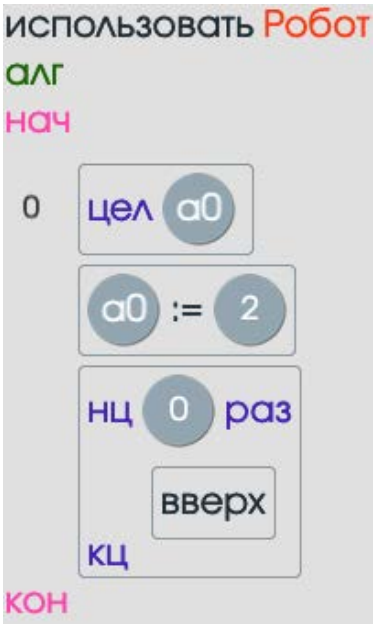


Рис. 7. Пример программы в среде ПиктоМир-К

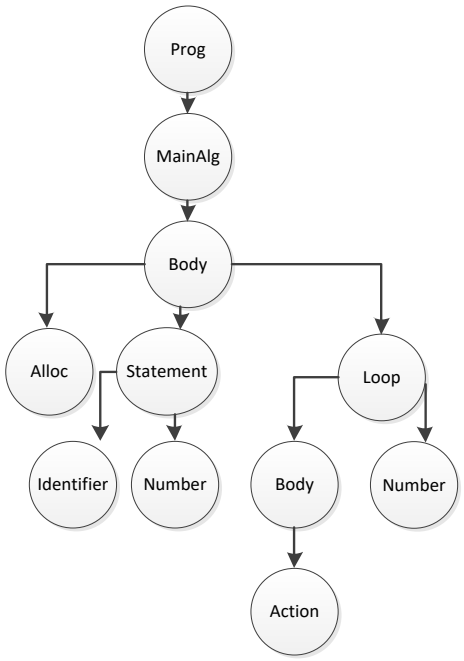


Рис. 8. Синтаксическое дерево программы

На рисунке 8 представлено синтаксическое дерево этой программы, где стрелками обозначены дочерние элементы вершины, а на самих узлах дерева написаны типы вершин синтаксического дерева, к которым они относятся.

В результате трансляции синтаксического дерева получаем следующий набор инструкций (см. табл. 3.).

Таблица 3. Набор инструкций

№	Инструкция	Аргументы	Значение
0	Jump	jumpLabel	“main”
1	Execute	paramQuantity isNative methodID	0 false “main”
2	Alloc	varType varName value	“int” “a0” 0
3	Push	Value	2
4	Store	varName fromReg	“a0” false
5	Push	Value	1
6	Fetch	Varname	“a0”
7	Start loop	reptype blockEndLabel	“ntimes” “lb_1”
8	Execute	isNative robot methodID	true robot “move_up”

		paramCount	0
9	End loop	robot jumpLabel	robot “lb_0”
10	Return	paramQuantity	0

Таблица 4. Маркеры перехода

Название ссылки	**Номер инструкции**
main	1
LB_0	8
LB_1	10

Выполнение программы начинается с начала главного алгоритма. В таблице 4 с маркерами перехода хранится это начало под именем main. На вершину стека укладывается номер инструкции, с которой начинается тело алгоритма. В таблице 5 показаны состояния регистра и стека во время выполнения программы. В данном примере выполняются действия объявления переменной, присвоения ей значения и пример работы цикла.

В таблице маркеров также хранятся метки начала и конца тела цикла. На каждой итерации проверяется условие окончания цикла. При невыполнении условия происходит переход к инструкции, следующей за телом цикла, она записана под маркером окончания цикла.

Таблица 5. Состояния регистра и стека во время выполнения программы

Позиция инструкции в списке	Номер инструкции	Регистр1	Стек
0	Jump 1	[]	[]
1	Execute main	[1]	[2]
2	Alloc int a0	[1]	[2, {type:”int”,name:”a0”, val:0}]
3	Push 2	[1]	[2, {type:”int”, name:”a0”,val:0}, 2]
4	Store a0	[1]	[2, {type:”int”, name:”a0”, val:2}]
5	Push 1	[1]	[2, {type:”int”, name:”a0”, val:2}, 1]
6	Fetch a0	[1]	[2, {type:”int”, name:”a0”, val:2}, 1, 2]
7	Start loop	[1]	[2, {type:”int”, name:”a0”, val:2}, 1, 2]
8	Execute move_up	[1]	[2, {type:”int”, name:”a0”, val:2}, 1, 2]
9	End loop	[1]	[2, {type:”int”, name:”a0”, val:2}, 2, 2]
10	Execute move_up	[1]	[2,

			{type:”int”, name:”a0”, val:2}, 2, 2]
11	End loop	[1]	[2, {type:”int”, name:”a0”, val:2}]
12	Return	[]	[]

Описанный метод компиляции и выполнения программы реализован на языке JavaScript [16] в ЦОС ПиктоМир-К.

Работа выполнена по теме 0065-2019-0010 «Разработка, реализация и внедрение семейства интегрированных многоязыковых сред программирования с автоматизированной проверкой заданий для учащихся образовательных организаций, ДОО, младшей, основной и старшей школы и студентов педагогических университетов» госзадания в отделе учебной информатики ФГУ ФНЦ НИИСИ РАН.

Building a Compiler-Interpreter for a Hybrid Text-Program Digital Educational Environment PictoMir-K

D.B. Agliamutdinova, M.V. Rayko, A.G. Leonov

Abstract. In the modern post-industrial world, students should become familiar with the basics of algorithmics at an early age, even before school. Digital educational environments should provide the teacher and student with a methodologically convenient mechanism for early teaching of algorithms, equipped with a pictographic programming language suitable for preschoolers who cannot yet read, or do not like to read. However, subsequently, the transition from pictographic to text programming becomes a difficult problem at school for young programmers. The hybrid digital educational system PiktoMir-K combines the simplicity of the usual "assembly" of programs from icons with a textual representation of the algorithm, and, in fact, is a bridge that allows the student to easily switch to school and production programming languages.

The article discusses the way of presentation and execution of programs designed in the digital educational environment PiktoMir-K, including a description of the programming language with syntax, semantics and alphabet. The data structure of the syntax tree of the program and its nodes is also described, the algorithm for creating a set of instructions corresponding to the syntax tree is considered, and the mechanism for executing these instructions using a virtual stack machine based on the execution machine of the PiktoMir environment is given.

Keywords: PiktoMir, digital educational environment, compiler, stack machine, programming

Литература

1. А.Г. Кушниренко, А.Г. Леонов. Роль программирования в непрерывном курсе информатики. «Международная конференция Математика и информационные технологии в нефтегазовом комплексе, посвящённая дню рождения великого русского математика академика П.Л. Чебышёва», Россия, Обнинск,14–19 мая 2019, Сургут, 23 мая 2019.Труды конференции (под ред. акад. В.Б. Бетелина). — ООО Порто-принт Самара, 2019, 62–63.

2. Материалы ТАСС, 8 декабря 2018, XVIII съезд «Единой России»: «Глава профильного комитета Думы считает нужным ввести информатику в дошкольную программу». [Электронный ресурс], URL: https://tass.ru/obschestvo/5888487 (дата обращения 15.11.2020).

3. А.Л. Семенов. Концептуальные проблемы информатики, алгоритмики и программирования в школе. «Вестник кибернетики», 2016, № 2(22).11–15. URL: https://elibrary.ru/download/elibrary_28842776_16454141.pdf (дата обращения 15.11.2020).

4. Головная страница проекта ПиктоМир на сайте ФГУ ФНЦ НИИСИ РАН. [Электронный ресурс], URL: https://www.niisi.ru/piktomir/ (дата обращения: 15.11.2020).

5. I.B. Rogozhkina, A.G. Kushnirenko. PiktoMir: Teaching Programming Concepts to Preschoolers with a New Tutorial Environment. “World Conference of Educational Technology and Researches”. – Procedia - Social and Behavioral Sciences, 28 (2011), 601 – 605. DOI: 10.1016/j.sbspro.2011.11.114

6. А.Д. Кисловская, А.Г. Кушниренко. Методика обучения алгоритмической грамоте дошкольни-

ков и младших школьников // Информационные технологии в обеспечении федеральных государственных образовательных стандартов: Материалы Международной научно-практической конференции. 16-17 июня 2014 года. - Елец: ЕГУ им. И. А. Бунина. Т. 2 (2014), 3-7. URL: https://elibrary.ru/item.asp?id=22284368 (дата обращения: 16.11.2020).

7. ДОПОЛНИТЕЛЬНАЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА технической направленности «Алгоритмика для дошколят», размещенная на сайте МБДОУ № 20 «Югорка» г. Сургута, [Электронный ресурс], URL: http://ds20.detkin-club.ru/editor/21/files/Образование/допобразование/17cc9a2cc5a5c633f69013cf62835fe6.pdf (дата обращения: 15.11.2020).

8. А.Г. Кушниренко, А.Г. Леонов, О.В. Собакинских, Л.В. Шибаева. Влияние обучения программированию на основе системы "ПИКТОМИР" на развитие психологических новообразований старших дошкольников. VIII Международная конференция «Воспитание и обучение детей младшего возраста» (ECCE 2019), России, Москва, 29 мая - 1 июня 2019 г. // Сборник «Воспитание и обучение детей младшего возраста». ISBN 978-5-19-011404-1

9. А.Г. Леонов, М.В. Райко, О.В. Собакинских, Н.В. Собянина. Результаты освоения годовой программы «Алгоритмика для дошколят» подготовительными группами муниципального ДОУ. «Труды НИИСИ РАН», Т. 10 (2020), № 5-6, XX-XX.

10. И.Н. Грибанова, Я.Н. Зайдельман, М.В. Райко Вводное занятие по алгоритмике в разновозрастной группе дошкольников и младших школьников. «Труды НИИСИ РАН», Т. 10 (2020), № 5-6, XX-XX.

11. В.Б. Бетелин, А.Г. Кушниренко, А.Г. Леонов. Основные понятия программирования в изложении для дошкольников. «Информатика и ее применения», 14(3): 55–61, 2020. DOI: http://dx.doi.org/10.14357/19922264200308

12. Н.О Бесшапошников, А.Г. Кушниренко, А.Г. Леонов, А.А. Малый. Проект двуязыковой пиктограммно-текстовой учебной среды программирования ПиктоМир-К. «XIV конференция Свободное программное обеспечение в высшей школе», Россия, Переславль, 25-27 января 2019 г. / Сборник тезисов. – М.: МАКС Пресс, 2019, 64–66.

13. А.Г. Кушниренко, А.Г. Леонов, Я.Н. Зайдельман, В.В Тарасова. Информатика. 7 класс. – М.: Дрофа, 2017. 180 с.

14. Н.О. Бесшапошников, А.Г. Леонов. Пиктограммный язык программирования «Пикто». «Вестник кибернетики», Т. 28 (2017), № 4, 173-180.

15. N. Wirth. The Programming Language Pascal. ″Acta Informatica″, Vol. 1 (1971), 35–63

16. ISO/IEC 22275:2018. Information technology — Programming languages, their environments, and system software interfaces — ECMAScript Specification Suite. [Электронный ресурс], URL: https://www.iso.org/standard/73002.html (дата обращения 15.11.2020).

Особенности реализации человеко-машинного интерфейса для детей младшего возраста в пропедевтических курсах по программированию

Н.О. Бесшапошников[1], М.С. Дьяченко[2], А.Г. Леонов[3], М.А. Матюшин[4], К.А. Мащенко[5], К.А. Прокин[6]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, nbesshaposhnikov@vip.niisi.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, mdyachenko@niisi.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, МГУ им. М.В.Ломоносова, Москва, Россия,

МПГУ, Москва, Россия, Государственный университет управления, Москва, Россия, dr.l@vip.niisi.ru;

[4]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, itsaprank@yandex.ru;

[5]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, МГУ им. М.В.Ломоносова, Москва, Россия, kirill010399@vip.niisi.ru;

[6]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, prokin@vip.niisi.ru;

Аннотация. Современные педагогические методики позволяют обучать основам алгоритмизации детей дошкольного возраста. По ряду причин, начиная работу с детьми дошкольного возраста, педагог должен стремиться выстроить процесс обучения программированию с минимальным индивидуальным использованием детьми компьютерной техники, включая и современные планшетные компьютеры. Если на начальном этапе объяснять понятия алгоритмизации можно с участием самих детей (один ребенок имитирует робота, а другой управляет им, отдавая команды), то для демонстрации автоматического выполнения программы уже нужны компьютерные средства и потому необходимо гладко стыковать компьютерные и некомпьютерные технологии. Для этого в учебный процесс предлагается ввести следующие элементы: создание программы путем механического манипулирования физическими объектами (пиктограммами в виде магнитов или кубиков), использование дополненной реальности при демонстрации работы программы в реальном мире. В статье рассмотрены технические средства, позволяющие ввести указанные элементы в процесс обучения детей дошкольного возраста.

Ключевые слова: ПиктоМир, дополненная реальность, пиктографическое программирование, дошкольное обучение алгоритмизации, человеко-машинный интерфейс, цифровой бот, программирование

1. Введение

Результаты проведенных в России и во всем мире отдельных экспериментов демонстрируют, что обучение детей алгоритмике можно начинать с 4х лет, хотя систематические освоение замкнутой системы научных понятий программирования, предложенное в работе [1], может быть завершено только в возрасте 6+ при прохождении курса алгоритмизации в выпускных группах детского сада. Ребенок поэтапно погружается в предмет алгоритмизации, на каждом этапе увеличивая степень взаимодействия с компьютерной техникой [2]. При этом дети должны минимум времени проводить с компьютером, включая планшетные компьютеры.

К особенностям детей дошкольного возраста можно отнести следующие:

- ребенок не умеет уверенно читать, но в ряде случаев может узнавать простые слова и выбирать знакомые изображения;
- ребенок может управлять роботами-игрушками голосовыми командами, но особенности речи детей не позволяют использовать решения по распознаванию речи, ориентированные на детей старшего возраста и взрослых;
- ребенок свободно оперирует физическими объектами, в частности, легко собирает мозаики-головоломки из кубиков и способен составлять программы управления роботами-игрушками из кубиков с нанесенными на их грани пиктограммами команд;
- ребенок обладает абстрактным мышлением,

обеспечивающим легкое восприятие дополненной реальности.

2. Цифровая образовательная среда ПиктоМир

До настоящего времени в процессе обучения дошкольников использовались специализированные программные средства, поддерживающие методы бестекстового программирования, например, система ПиктоМир [3, 4, 5].

Сколь бы простой не была подобная система, до начала работы с ней преподавателю необходимо познакомить ребенка с терминологией, объяснить назначение элементов интерфейса и базовые принципы работы системы [6]. При этом дети могут с разной скоростью осваивать новый материал, поэтому преподавателю приходится многократно отвечать на однотипные вопросы по работе с системой или выявлять детей, которые испытывают трудности в освоении системы, но стесняются задавать вопросы. Ниже рассматриваются особенности реализации человеко-машинного интерфейса в курсе обучения алгоритмизации детей дошкольного возраста и способы, позволяющие снять часть нагрузки с преподавателя и улучшить освоение материала за счет персонализации процесса обучения [7].

В качестве решения, позволяющего учесть приведенные выше особенности, был выбран диалоговый интерфейс, реализующий общение ребенка со встроенным ботом — экранным персонажем системы ПиктоМир. Экранный персонаж «Бот», согласно легенде, является сверстником ребенка, который вместе с ним шаг за шагом постигает основы алгоритмизации в ПиктоМире. В учебном процессе Бот позволяет снять с преподавателя часть нагрузки, связанной с ответами на простые вопросы по интерфейсу используемой программной системы ПиктоМир. Основным методом общения с Ботом является речевое общение. Если ребенку не удается наладить речевой контакт с Ботом (проблема с дикцией, неумение сформулировать, в чем заключается проблема, забывание терминов), используется вариант ввода вопроса с использованием экрана игры (см. рис. 1).

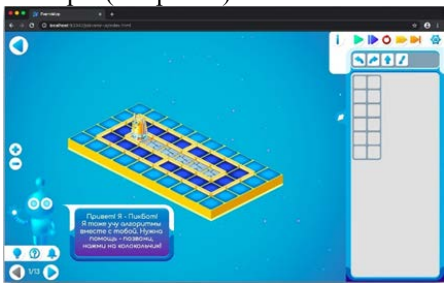


Рис. 1. Пример интерфейса взаимодействия бота с ребенком

3. Применение элементов ИИ в образовательном процессе дошкольников

Но основное время дети проводят за составлением программ. В этом случае традиционные форматы взаимодействия через экранный интерфейс приложения являются менее предпочтительными, чем использование физических объектов. Демонстрируя процесс составления программы в системе ПиктоМир, вместо демонстрации работы с программной системой на сенсорном экране электронной доске, преподаватель может использовать магнитные карточки, механически размещая их в нужном порядке на магнитной доске, а дети могут составлять программы, механически выкладывая на своем столе кубики, на грани которых нанесены пиктографические отображения команд. Для передачи компьютеру составленной таким образом программы достаточно сфотографировать составленную конфигурацию кубиков. Сфотографированная конфигурация будет распознана специальной компонентой системы ПиктоМир и преобразована в программу, которая далее может быть исполнена системой ПиктоМир [8] (см. рис. 2). Преимущества подобной методики «физического» процесса составления программы состоят в том, что не требует освоение интерфейса для манипуляции пиктограммами команд на экране и, что очень важно, период экранной работы ребенка в приложении, продолжительность которого ограничивается санитарными нормами, сводится к короткому просмотру процесса выполнения составленной из кубиков программы. А во время более продолжительной работы по составлению программы экран не задействован.

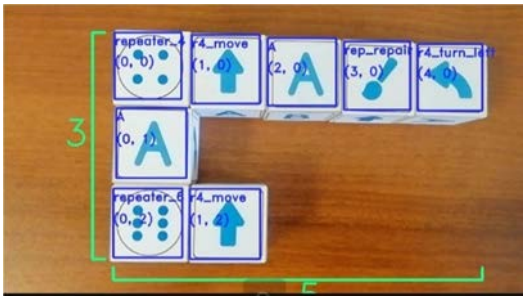


Рис. 2. Пример составленной из кубиков программы

Для усиления мотивации детей путем создание визуально привлекательных обстановок и для демонстрации возможностей современных технологий связывания виртуального и реального миров, может быть использован режим дополненной реальности. В этом режиме после составления программы ребенок может увидеть, как виртуальный робот выполняет программу на

реальном игровом поле, только что собранном детьми из сочленяемых ковриков на полу в реальном интерьере игровой комнаты (см. рис. 3). Такая технология хорошо воспринимается детьми и правильно ими интерпретируется.

С точки зрения организаторов учебного процесса подобная технология привлекательна тем, что не требует приобретения дорогостоящих и сложных в обслуживании движущихся радиоуправляемых автономных роботов (аккумуляторы виртуальных роботов из дополненной реальности пожаробезопасны и всегда заряжены на 100% :-).

Рис. 3. Пример отображения робота в режиме дополненной реальности

С целью улучшения контроля за процессом усвоения учебного материала детьми, Бот может использовать визуальный канал взаимодействия. Например, регистрируя эмоциональный отклик ребенка. Регистрация эмоционального отклика пользователя при общении через Интернет в последнее время является активной областью исследований, см., например, [9]. Большинство реализаций методов анализа эмоционального отклика, см. [10], [11], в том или ином виде использует методы машинного обучения, привлекая для тренировки большие открытые наборы данных [12]. Одним из способов реализации данного механизма является распознавание эмоции по изображению лица ребенка. Данный способ имеет определенные преимущества по сравнению с описанными в [9] носимыми датчиками, поскольку для детектирования требуется лишь камера, которая доставляет минимальный дискомфорт в использовании, если таковой вообще имеется. Техническая реализация алгоритма заключается в использовании сверточных архитектур нейронных сетей. В ходе тренировки решается задача классификации - разделение классов различных эмоциональных выражений лица. Для увеличения точности лицо может быть предварительно кадрировано с помощью хорошо изученных алгоритмов [13]. Данная технология позволяет в реальном времени получать от ребенка отклик о результатах процесса обучения, который, будь он положительным или отрицательным, затем может быть использован с целью корректировки траектории, проходимым обучаемым в данном курсе. Именно за счет возможности оперативной реакции, данный подход, несомненно, является одной из наиболее перспективных для практического применения.

3. Заключение

В результате выполненного исследования реализован встроенный в ПиктоМир бот, обладающий следующими возможностями:

- Бот методически верно подает материал в ответ на задаваемые вопросы.
- Бот адаптивен и оценивает, когда ребенку требуется что-то повторить, напомнить, объяснить другим способом.
- Бот предоставляет ответы на вопросы по интерфейсу системы, демонстрирует возможности системы в наглядной форме.

В дополнение к боту реализованы механизмы распознавания программ и дополненной реальности.

По результатам анализа обратной связи от преподавателей отмечено, что

- наличие диалогового интерфейса,
- распознавание программ, составленных из физических объектов, и
- использование дополненной реальности помогают в образовательном процессе даже при обучении детей дошкольного возраста.

В продолжении данных исследований планируется изучить методы удержания внимания ребенка, оценки уровня восприятия и усталости ребенка.

Работа выполнена при поддержке гранта РФФИ № 19-29-14057.

Features of the Implementation of a Human-Machine Interface for Young Children in Propaedeutic Programming Courses

N. Besshaposhnikov, M. Dyachenko, A. Leonov, M. Matyushin, K. Mashchenko, K. Prokin

Abstract. Modern pedagogical methods support teaching preschoolers the basics of algorithmization. For a number of reasons, starting work with children of preschool age, the teacher should strive to build the process of teaching programming with minimal individual use of computer technology by children, including modern tablet computers. If at the initial stage it is possible to explain the concepts of algorithmization with the participation of the children themselves (one child imitates a robot, and the other controls it by giving commands), then to demonstrate the automatic execution of a program, computer tools are already needed and therefore it is necessary to smoothly combine computer and non-computer technologies. To do this, it is proposed to introduce the following elements into the educational process: creating a program by mechanical manipulation of physical objects (pictograms in the form of magnets or cubes), using augmented reality when demonstrating the program's work in the real world. The article discusses the technical means that allow the introduction of these elements into the learning process of preschoolers.

Keywords: PictoMir, augmented reality, pictographic programming, preschool learning algorithms, human-machine interface, digital bot, programming

Литература

1. В.Б. Бетелин, А.Г. Кушниренко, А.Г. Леонов. Основные понятия программирования в изложении для дошкольников // Информатика и ее приложения, 2020. Т. 14. Вып. 3. С. 56-62. DOI: 10.14357/19922264200308

2. А.Г. Кушниренко, А.Г. Леонов, М.В. Райко, Методические указания по проведению цикла занятий «Алгоритмика» в подготовительных группах дошкольных образовательных учреждений с использованием свободно распространяемой учебной среды ПиктоМир. [Электронный ресурс]// Свободно распространяемый методический материал на сайте ФГУ ФНЦ НИИСИ РАН. URL:https://www.niisi.ru/piktomir/Алгоритмика для дошкольников. 19.09.2019.pdf (дата обращения 02.11.2020).

3. А.Г. Кушниренко, А.Г. Леонов. Роль программирования в непрерывном курсе информатики. «Международная конференция Математика и информационные технологии в нефтегазовом комплексе, посвящённая дню рождения великого русского математика академика П.Л. Чебышёва», Россия, Обнинск, 14–19 мая 2019.

4. Н.О. Бесшапошников, А.Г. Кушниренко, А.Г. Леонов, К.А. Прокин. Технологические инновации меняют методику курса Алгоритмика для дошкольников. «Воспитание и обучение детей младшего возраста: VIII Международная конференция (ECCE 2019)», Россия, Москва, 29 мая — 1 июня 2019, URL: https://www.niisi.ru/piktomir/m2019.pdf (дата обращения 02.11.2020).

5. I.B. Rogozhkina, A.G. Kushnirenko. PiktoMir: Teaching Programming Concepts to Preschoolers with a New Tutorial Environment. “World Conference of Educational Technology and Researches”. – Procedia - Social and Behavioral Sciences 28 (2011), 601 – 605. DOI: 10.1016/j.sbspro.2011.11.114

6. Н.О. Бесшапошников, А.Г. Леонов. Пиктограммный язык программирования «Пикто». «Вестник кибернетики», Т. 28 (2017), № 4, 173-180.

7. Н.О. Бесшапошников, А.Г. Леонов, А.А. Прилипко. Цифровизация образования – Новые возможности управления образовательными треками. «Вестник кибернетики», Т. 30 (2017), № 3, 154-161.

8. Н.О. Бесшапошников, М.С. Дьяченко, М.А. Кузьменко и др. Автоматическая разметка кадров видеопотока для машинного обучения «Труды НИИСИ РАН», Т. 9 (2019), № 6, 118–122. URL: https://www.niisi.ru/tr/2019_T9_N6.pdf. (дата обращения 02.11.2020).

9. G.Price. Tapping Into the Emotional Internet. “TechCrunch”, 2015 [Электронный ресурс]
URL: https://techcrunch.com/2015/08/23/tapping-into-the-emotional-internet/ (дата обращения 02.11.2020).

10. Y. Miyakoshi and S. Kato. Facial emotion detection considering partial occlusion of face using

Bayesian network. “2011 IEEE Symposium on Computers & Informatics”, Kuala Lumpur, 2011, 96–101, DOI: 10.1109/ISCI.2011.5958891.

11. B. Schuller, G. Rigoll and M. Lang. Hidden Markov model-based speech emotion recognition. “"2003 International Conference on Multimedia and Expo. ICME '03. Proceedings (Cat. No.03TH8698)”, Baltimore, MD, USA, 2003, 1–401, DOI: 10.1109/ICME.2003.1220939.

12. Affective, сайт компании [Электронный ресурс]// URL: https://www.affectiva.com (дата обращения 02.11.2020).

13. K. Zhang, Z. Zhang, Z. Li and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. ″IEEE Signal Processing Letters″, (2016), 23(10), 1499–1503.

Вводное занятие по алгоритмике в разновозрастной группе дошкольников и младших школьников

И.Н. Грибанова[1], Я.Н. Зайдельман[2], М.В. Райко[3]

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

[2]ФГУ ФНЦ НИИСИ РАН, г. Москва, Россия, yz@pereslavl.ru

[3]ФГУ ФНЦ НИИСИ РАН, г. Москва, Россия, mila.rayko@gmail.com

Аннотация. Современные педагогические методики и «умные» учебные пособия позволяют буквально в течение одного занятия продемонстрировать детям возможность пультового и программного управления реальным роботом-игрушкой и пробудить интерес к научно-технической тематике у детей самого разного возраста. В настоящей статье обобщен накопленный авторами опыт проведения подобных вводных занятий.

Ключевые слова: алгоритмика, робот, команда, программа, пульт, компьютер, ПиктоМир, Ползун

1. Введение

В статье рассматривается сценарий вводного занятия по алгоритмике в разновозрастной группе дошкольников и младших школьников с использованием цифровой образовательной среды ПиктоМир [1], методику работы с которой ФГУ ФНЦ НИИСИ РАН развивает в течение последних лет [2]. Такое занятие можно проводить на мероприятиях, направленных на популяризацию науки (например, научно-технические фестивали, дни науки), в образовательных организациях при проведении дней открытых дверей, в других случаях, когда необходимо за короткое время продемонстрировать детям сущность алгоритмического подхода и вызвать у них интерес к этой деятельности [3].

Сценарий занятия рассчитан на работу с группой из 6-8 детей в возрасте 5-10 лет. Продолжительность занятия от 30 до 60 минут. В сценарии указаны основные этапы занятия, каждый из которых можно провести чуть быстрее или чуть медленнее, рассмотреть отдельные вопросы более или менее подробно. Таким образом, на основе одного схематического сценария можно провести разные по наполнению занятия в зависимости от возраста детей и выделенного на занятие времени.

Для проведения занятия используется следующее оборудование: мягкие фигурки, изображающие роботов (Вертун, Двигун, Тягун), магнитная доска, комплект магнитов с пиктограммами команд роботов, реальный механический робот Ползун, разноцветные сочленяемые коврики для сборки игрового поля, по которому будет по командам перемещаться робот Ползун, комплект планшетов с установленной и настроенной системой ПиктоМир [1].

Из перечисленного выше оборудования строго обязательны только планшеты с ПиктоМиром. Остальные элементы желательны, но провести занятие можно даже при их отсутствии. Например, если есть доска, но нет магнитов с пиктограммами, можно просто рисовать команды на доске, а если наоборот, есть магниты, но нет доски, можно выкладывать пиктограммы на столе.

Занятие состоит из нескольких этапов. Продолжительность каждого этапа зависит от возраста детей (с младшими нужно более подробно рассматривать простые элементы, старшим можно давать более сложные задания) и общего времени, выделенного на занятие. В рамках каждого этапа можно пропускать отдельные упражнения или вставлять дополнительные, важно соблюдать общую последовательность этапов и их содержание.

2. Знакомство с роботами. Непосредственное управление

На этом этапе мы знакомим детей с понятием робота (исполнителя) как устройства, способного воспринимать и выполнять ограниченный набор команд, и демонстрируем непосредственное управление исполнителем.

Занятие начинается с обсуждения понятия «робот-исполнитель».

Современные дети хорошо знакомы с разнообразными управляемыми устройствами. Пульты управления телевизором или другой техникой, радиоуправляемые игрушки привычны

даже для малышей. Поэтому понимание того, что какое-то устройство в принципе способно воспринимать и исполнять команды, не вызывает сложностей. Но важно сразу отметить существенные особенности исполнителей, над которыми дети часто не задумываются.

Во-первых, набор команд каждого исполнителя строго фиксирован. У исполнителя есть система команд – список всех команд, которые он способен выполнить. У реальных исполнителей система команд может быть довольно большой (на пульте управления телевизором много кнопок), но она все равно не бесконечна. Исполнитель может понимать только те команды и выполнять только те действия, которые входят в его систему команд.

Во-вторых, исполнитель не думает и не решает задачи, он только выполняет команды. Если мы дали исполнителю неверный приказ, он его исполнит, потому что исполнитель ничего знает о том, что мы задумали и для чего отдаем этот приказ.

На занятии мы показываем детям мягкие фигурки, изображающие роботов Вертуна, Двигуна и Тягуна (см рис.1.а, 1.б, 1.в), и рассказываем про этих роботов. Каждый из них передвигается по клетчатому полю.

Рис.1.а. Вертун

Рис.1.б. Двигун

Рис.1.в. Тягун

У Вертуна это покрытая плитками площадка космодрома, у Двигуна и Тягуна – склад. Задача Вертуна – восстанавливать (закрашивать) плитки, поврежденные при взлете ракет. (Если есть время и техническая возможность, можно показать короткий видеоролик о взлете реальной ракеты). Задача Двигуна и Тягуна – расставить на нужные места предметы на складе.

У всех трёх роботов одинаковые команды движения по клетчатому полю: вперёд (робот переходит на следующую клетку), налево (поворот на месте на 90 градусов налево) и направо (поворот на месте на 90 градусов направо). Эти команды обозначаются простыми пиктограммами (см. рис. 2.а, 2.б, 2.в, 2.г, 2.д). У Вертуна и Тягуна есть еще дополнительные команды, связанные с выполняемыми задачами: у Вертуна – команда закрасить, у Тягуна – тянуть, но эти команды можно пока не рассматривать.

Рис.2.а. ***вперёд***

Рис.2.б. ***налево***

Рис.2.в. ***направо***

Рис.2.г. ***закрасить***

Рис.2.д. ***тянуть***

Далее управление роботами рассматривается на примере. Необходимо провести Вертуна по лабиринту (см.рис.3) (лабиринт можно нарисовать на доске или вывести его изображение на экран проектора). Для этого нужно последовательно давать роботу команды.

Рис.3. Лабиринт с роботом Вертун

Если позволяет время, можно предложить детям самостоятельно решить эту задачу. Для этого

дети разбиваются на пары. Каждая пара получает напечатанный на бумаге лабиринт и фигурку робота (см. рис. 4). Один ребенок в паре выполняет роль командира (отдает команды), другой – робота (выполняет полученные команды, перемещает фигурку).

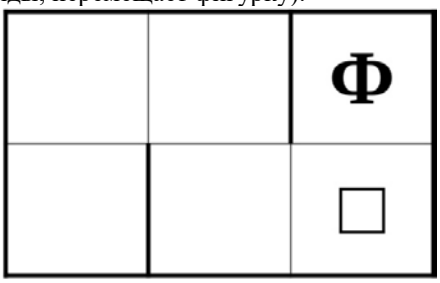


Рис.4. Схема-лабиринт

После самостоятельного решения (или сразу, если на самостоятельное решение нет времени) задача общими усилиями решается на доске.

В процессе решения команды отдаются по одной и сразу же выполняются. Преподаватель обсуждает с детьми, какую команду надо дать следующей. Робот (магнитная фигурка) передвигается по лабиринту на доске, преподаватель может двигать робота сам или поручить это ребёнку (возможно, нескольким детям по очереди). После выполнения каждой команды она обязательно записывается. Удобнее всего фиксировать команды с помощью магнитных пиктограмм, но, если магнитов нет, можно просто рисовать пиктограммы на доске.

После того, как весь лабиринт пройден, на доске остается запись всех выполненных команд (см. рис. 5).

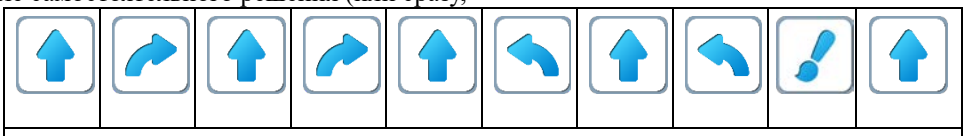


Рис.5. Программа, выложенная из магнитных пиктограмм

3. Программное управление

Содержание этого этапа – объяснение детям принципа программного управления. Между командиром (человеком) и исполнителем (роботом) появляется посредник – компьютер.

Рассмотрим задачу, которую мы только что решили. Предположим, что нужно еще раз провести робота по тому же самому лабиринту. Чтобы сделать это, нужно еще раз дать роботу те команды, которые записаны у нас на доске. Но есть одно важное отличие: нам теперь не надо думать над тем, какую команду дать следующей, можно просто смотреть на готовый список, выбирать из него очередную команду и давать роботу соответствующий приказ.

Получается, что при повторном управлении можно не думать, достаточно просто по очереди давать роботу команды из уже известного списка. А если думать не обязательно, эту работу может делать не человек, а какой-нибудь специально обученный исполнитель, который не думает, но может давать роботу команды из заданного списка. Если лабиринт нужно проходить много раз (например, робот должен каждый день проходить по одному и тому же маршруту), то человеку может надоесть каждый раз выполнять одни и те же действия, человек может устать и совершить ошибку, а исполнитель не устает и не ошибается.

Такой исполнитель, способный управлять роботами, называется *компьютер*, список команд, которые надо дать роботу, чтобы решить задачу – *программой*, а способ управления, при котором роботом управляет компьютер, берущий команды из заранее составленной программы, называется *программным управлением*. [6]

Откуда берется программа для программного управления? Чтобы составить программу, нужно думать, поэтому программу составляет человек, который называется *программистом*.

Мы с вами только что составили свою первую программу, то есть выполнили работу программиста. Но мы сейчас совмещали составление программы и непосредственное управление роботом. На самом деле программист чаще всего не управляет роботом сам. Программист мысленно представляет себе, как робот будет выполнять команды, подбирает необходимые команды для решения задачи, составляет из них программу и передает ее компьютеру. Компьютер выполняет программу и в процессе выполнения передает записанные команды роботу. Робот получает команды от компьютера, выполняет их и в результате совершает необходимые действия.

4. Программная система ПиктоМир

На этом этапе дети знакомятся с программной системой ПиктоМир. Каждый ребенок получает планшет, на котором установлена система и загружены необходимые для занятия задания. Для экономии времени можно заранее запустить первое задание, чтобы дети увидели его сразу

после включения планшетов.

Преподаватель показывает детям инструменты управления в ПиктоМире: как составить программу (перенести необходимые команды с полки команд в шаблон программы), как исправить возможные ошибки, как выполнить полученную программу.

Дети под руководством преподавателя решают 1-2 несложные задачи

5. Робот Ползун

Этот этап занятия – факультативный, его можно пропустить при недостатке времени или отсутствии реального Ползуна. Но управление Ползуном – очень зрелищный и привлекательный этап, поэтому если такая возможность есть, нужно обязательно показать его детям [4], [5].

Ползун (см. рис. 6.а) – реальный робот, который может воспринимать команды и выполнять их. Команд у Ползуна всего три – это уже знакомые нам команды движения вперед, направо, налево. Ползун передвигается по игровому полю, составленному из специальных ковриков (см. рис. 6.б), но, если ковриков нет, можно запускать Ползуна просто на полу или на большом демонстрационном столе.

Рис.6.а. Реальный робот Ползун

Рис.6.б. Игровое поле для реального Ползуна

Ползуном можно управлять в непосредственном режиме с помощью пульта, преподаватель показывает детям, как это делается.

В ПиктоМире есть программная модель Ползуна. Можно составить программу и управлять Ползуном на экране так же, как мы делали это с Вертуном. Дети самостоятельно или с помощью преподавателя составляют на планшетах программу управления Ползуном.

Теперь возьмем один планшет с готовой программой и включим реального Ползуна. Мы видим, что на экранном изображении Ползуна красная лампочка стала зеленой. Это значит, что экранный Ползун установил связь с реальным. Запустим на выполнение программу на планшете. Компьютер по очереди передает роботу команды программы, экранный Ползун выполняет их на экране, а реальный – на коврике! При этом мы слышим, как именно компьютер передает команды: каждой команде соответствует свой звук, Ползун слышит эти звуки и, подчиняясь им, движется вперед и выполняет повороты.

Этот наглядный пример очень важен для мотивации и для лучшего понимания сути программного управления: мы видим, как компьютер управляет *реальным* (существующим отдельно от компьютера, не на экране) роботом по заранее составленной программе.

6. Самостоятельная работа с планшетами

В конце занятия дети получают возможность самостоятельно решить несколько несложных задач в системе ПиктоМир. Подготовленный комплект заданий включает по два задания для каждого из рассмотренных роботов (Вертун, Двигун, Тягун, Ползун). Старшие дети обычно справляются с заданиями самостоятельно, младшим может потребоваться помощь преподавателя. Составленные программы для Ползуна можно исполнить, управляя не только нарисованным роботом на экране, но и реальным роботом на игровом поле из сочленяемых ковриков.

В конце занятия дети получают адрес сайта ПиктоМир [1], где можно более подробно познакомиться с системой и продолжить решать задачи через веб-интерфейс или загрузив свободно распространяемое приложение «ПиктоМир».

Работа выполнена по теме 0065-2019-0010 «Разработка, реализация и внедрение семейства интегрированных многоязыковых сред программирования с автоматизированной проверкой заданий для учащихся образовательных организаций, ДОО, младшей, основной и старшей школы и студентов педагогических университетов» госзадания в отделе учебной информатики ФГУ ФНЦ НИИСИ РАН.

Приложение

Пример заданий вводного занятия

Мир «Вводное занятие по алгоритмике» [7], который доступен на сайте ПиктоМир, состоит из трех игр. Так как группа может быть разновозрастной, то необходим набор заданий разного уровня сложности. Первая игра «Знакомство с Роботами» предназначена для знакомства с игрой ПиктоМир, задачи в этой игре дети решают под руководством преподавателя. Две следующие игры «Простые задачи» и «Задачи посложнее» для самостоятельного (или с помощью преподавателя) решения.

Задание1. В этом задании знакомимся с роботом Вертун. Роботу нужно починить помеченную клетку и добраться до финиша (см. рис. 1.а и 1.б).

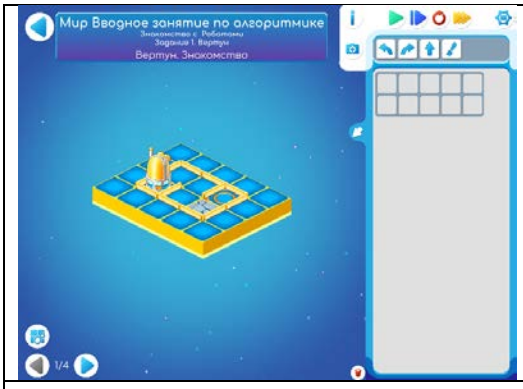


Рис. 1.а. Шаблон программы задания с роботом Вертун

Рис. 1.б. Решение к заданию с роботом Вертун

Задание 2. В этом задании знакомимся с роботом Ползун. Роботу нужно пройти по всем клеткам с цифрами от «1» до «3» и закончить маршрут в клетке с «Х» (финиш) (см. рис. 2.а и 2.б).

Рис. 2.а. Шаблон программы задания с роботом Ползун

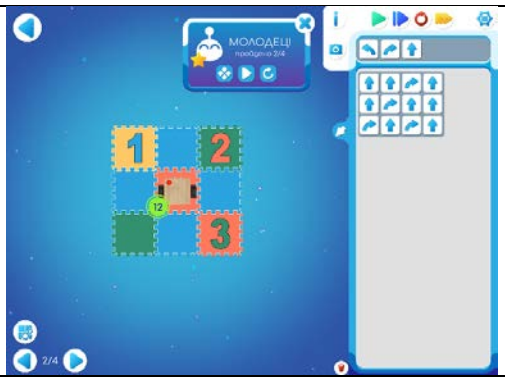


Рис. 2.б. Решение к заданию с роботом Ползун

Задание 3. В этом задании знакомимся с роботом Двигун. Роботу нужно расставить по местам грузы: бочку на зеленую клетку с белым кружочком, ящик на зеленую клетку с белым крестиком (см. рис. 3.а и 3.б).

Рис. 3.а. Шаблон программы задания с роботом Двигун

Рис. 3.б. Решение к заданию с роботом Двигун

Задание 4. В этом задании знакомимся с роботом Тягун. Задача этого робота растащить по нужным местам грузы (см. рис. 4.а и 4.б).

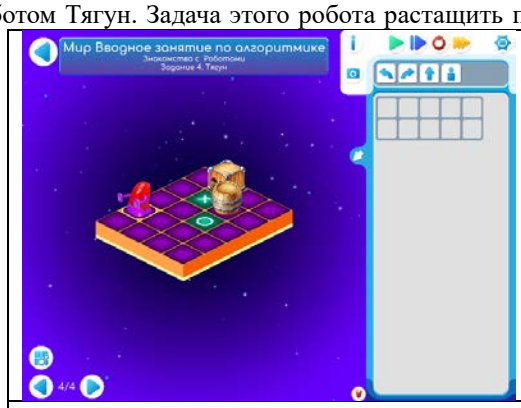


Рис. 4.а. Шаблон программы задания с роботом Тягун

Рис. 4.б. Решение к заданию с роботом Тягун

Игра «Простые задачи»

Задание1. Роботу Вертуну нужно починить все клетки (см. рис. 5.а и 5.б).

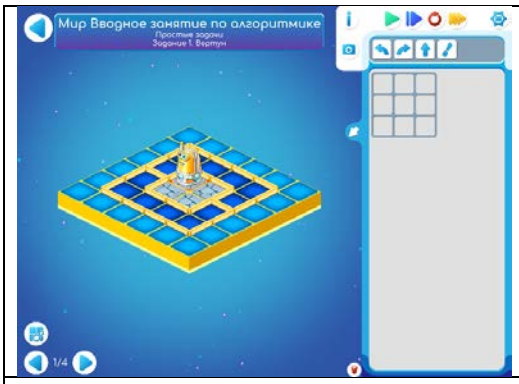


Рис. 5.а. Шаблон программы задания с роботом Вертун

Рис. 5.б. Решение к заданию с роботом Вертун

Задание 2. Роботу Ползуну нужно пройти по всем клеткам с цифрами от «1» до «3» и закончить маршрут в клетке с «X» (финиш) (см. рис. 6.а и 6.б).

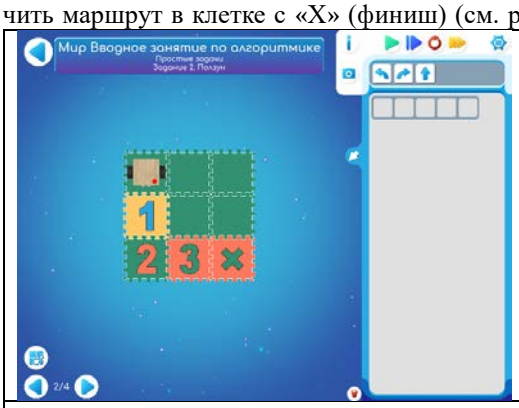


Рис. 6.а. Шаблон программы задания с роботом Ползун

Рис. 6.б. Решение к заданию с роботом Ползун

Задание 3. Роботу Двигуну нужно поставить ящик на зеленую клетку с белым крестиком и добраться до финиша (см. рис. 7.а и 7.б).

Рис. 7.а. Шаблон программы задания с роботом Двигун

Рис. 7.б. Решение к заданию с роботом Двигун

Задание 4. Задача робота Тягуна растащить по нужным местам грузы (см. рис. 8.а и 8.б).

Рис. 8.а. Шаблон программы задания с роботом Тягун

Рис. 8.б. Решение к заданию с роботом Тягун

Игра «Задачи посложнее»

Задание 1. Роботу Вертуну нужно починить помеченные клетки и добраться до финиша (см. рис. 9.а и 9.б).

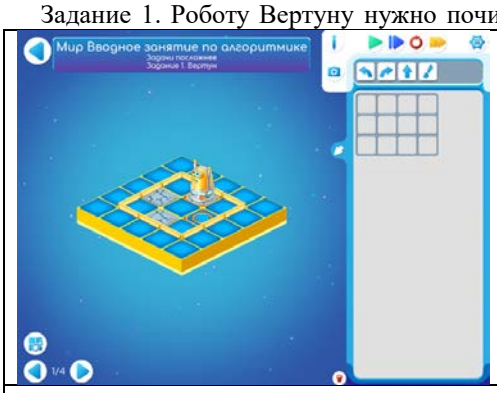


Рис. 9.а. Шаблон программы задания с роботом Вертун

Рис. 9.б. Решение к заданию с роботом Вертун

Задание 2. Роботу Ползуну нужно пройти по всем клеткам с цифрами от «1» до «4» и вернуться на старт (см. рис. 10.а и 6.б).

Рис. 10.а. Шаблон программы задания с роботом Ползун.

Рис. 10.б. Решение к заданию с роботом Ползун.

Задание 3. Роботу Двигуну нужно раздвинуть ящики и прийти к финишу (см. рис. 11.а и 11.б).

Рис. 11.а. Шаблон программы задания с роботом Двигун.

Рис. 11.б. Решение к заданию с роботом Двигун.

Задание 4. Задача робота Тягуна растащить грузы по нужным местам (см. рис. 12.а и 12.б).

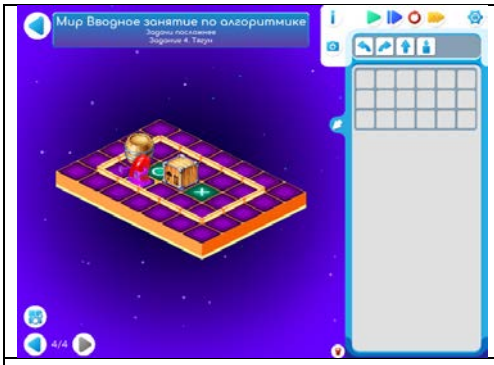


Рис. 12.а. Шаблон программы задания с роботом Тягун.

Рис. 12.б. Решение к заданию с роботом Тягун.

Introductory Lesson on Algorithmic in a Multi-Age Group of Preschoolers and Primary Schoolchildren

I.N. Gribanova, M.V. Rayko, Ya.N. Zaydelman

Abstract. Modern pedagogical methods and "smart" training kits allow children to demonstrate the possibility of remote and software control of a real robot toy and awaken interest in scientific and technical topics in children of all ages literally during one lesson. This article summarizes the authors ' experience in conducting such introductory classes.

Keywords: algorithmic, robot, command, program, remote, computer, Pictomir, Slide

Литература

1. Головная страница проекта ПиктоМир на сайте ФГУ ФНЦ НИИСИ РАН [Электронный ресурс]. URL: https://www.niisi.ru/piktomir/ (дата обращения: 16.11.2020).

2. А.Д. Кисловская, А.Г. Кушниренко. Методика обучения алгоритмической грамоте дошкольников и младших школьников // Информационные технологии в обеспечении федеральных государственных образовательных стандартов: Материалы Международной научно-практической конференции. 16-17 июня 2014 года. - Елец: ЕГУ им. И. А. Бунина. Т. 2 (2014), 3-7. URL: https://elibrary.ru/item.asp?id=22284368 (дата обращения: 16.11.2020).

3. А.Г. Кушниренко, А.Г. Леонов, И.Н. Грибанова, М.В. Райко. Проведение цикла занятий «Алгоритмика» в летнем лагере для дошкольников и младшеклассников. «Труды НИИСИ РАН», Т. 8 (2018), № 4, 181-185

4. А.Г. Кушниренко, А.Г. Леонов. Элективный курс для младшеклассников «Управляем настоящим Роботом». Новое слово в науке и практике: гипотезы и апробация результатов исследований, 2014, с. 105-109

5. А.Г. Кушниренко, А.Г. Леонов, А.А. Левин, А.А. Малый. Музыкальное управление учебным роботом в курсе «Алгоритмика для дошкольников». VIII Международная конференция "Воспитание и обучение детей младшего возраста" 29 мая-1 июня 2019 (ECCE 2019), Москва, МГИМО МИД России, Россия, 29 мая - 1 июня 2019

6. Н.О. Бесшапошников, А.Г. Леонов. Пиктограммный язык программирования «Пикто». «Вестник кибернетики». Т. 28 (2017), № 4, 173-180.

7. Методические указания и комплект ПиктоМир-Игр для проведения вводного занятия по алгоритмике в разновозрастной группе дошкольников и младших школьников на сайте ФГУ ФНЦ НИИСИ РАН [Электронный ресурс]. URL: https://www.niisi.ru/piktomir/method.htm (дата обращения: 16.11.2020).

Необходимость включения исполнителей со сложным поведением в курс бестекстового программирования для дошкольников и первоклассников

А.Г. Кушниренко[1], К.А. Мащенко[2], А.Е. Орловский[3], Н.А. Серебрицкая[4]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, agk_mail.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, kirill010399@vip.niisi.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, 4orlovskiy@niisi.ru;

[4]ФГУ ФНЦ НИИСИ РАН, Москва, serebr@vip.niisi.ru

Аннотация. В настоящей статье описаны доводы в пользу включения в завершающую часть курса бестекстового программирования для дошкольников практикума по составлению алгоритмов управления роботами-исполнителями со сложным поведением, приведен пример одного такого сложного робота, работающего в связке с пассивными роботами-партнерами, и показано, что введение подобных роботов позволяет разнообразить и усложнить набор заданий по программированию, доступных дошкольникам и первоклассникам.

Ключевые слова: Алгоритмика, робот, исполнитель команд, система команд, пиктограмма, программирование, дошкольник, первоклассник, вводный курс, ПиктоМир, сложное поведение

1. Целесообразность освоения роботов со сложным поведением в бестекстовом курсе программирования

Система бестекстового программирования ПиктоМир и подготовленные в ее рамках наборы заданий по программированию роботов и других исполнителей команд показали свою практическую применимость и достаточно высокую эффективность в ходе массового внедрения курса «Алгоритмика для дошколят» в системе дошкольного образования г. Сургута [1]. Система и ее методическое наполнение оказались эффективными и при продолжении знакомства с азами программирования в начальной школе. Опыт эксплуатации системы ПиктоМир и наборов заданий для курса «Алгоритмика для первоклассников» [2], продолжающего курс для дошкольников, показал, что на завершающих этапах бестекстового изучения программирования детям становится доступно понимание усложненных игровых обстановок, в которых действуют роботы-исполнители с усложненным поведением. Эта способность работы со сложными структурами, приобретаемая по достижении возраста 6-7 лет, должна быть, по нашему мнению, использована для ускорения ***познавательного и речевого развития*** детей.

Федеральный государственный образовательный стандарт дошкольного образования [3] рекомендует в целях познавательного и речевого развития освоение детьми сюжетно-ролевых игр и игр с правилами. Важным развивающим элементом годового курса «Алгоритмика для дошколят» и полугодового курса «Алгоритмика для первоклассников» является практическое освоение систем команд (правил работы) различных движущихся роботов и неподвижных исполнителей. Освоение этих правил в группах дошкольников проводится с помощью игр-имитаций. Каждая такая имитация может рассматриваться как подвижная игра по определенным правилам в некоторой игровой обстановке. При этом возникает вопрос о выборе оптимального уровня сложности правил, определяющих структуру обстановки и поведение роботов. В уже опробованных версиях курсов «Алгоритмика для дошколят» и «Алгоритмика для первоклассников» выбор был сделан в направлении минимизации сложности. Структура игровой обстановки и системы команд роботов и исполнителей цифровой среды «ПиктоМир», задействованных в начальных курсах [1], [2], до настоящего времени намеренно подбирались простыми логически и геометрически. В этих курсах используются роботы *Вертун*, *Двигун*, *Тягун* и *Ползун*, перемещающиеся по клетчатым игровым полям. Мы будем называть этих роботов ***простыми***.

Каждый простой робот перемещается по игровому полю, состоящему из квадратных клеток, и занимает на поле ровно одну клетку. Между

клетками поля могут размещаться стены. На некоторых клетках могут быть размещены ***стоянки*** или ***базы*** для долговременного нахождения роботов и других объектов игрового поля. В заданиях на программирование простых роботов обычно требуется, чтобы по завершении программы робот и другие объекты оказались размещены на стоянках. Каждый простой робот способен выполнять небольшое число интуитивно очевидных команд. Игровая обстановка также определяется простыми «одноклеточными» правилами: каждый объект игровой обстановки простого робота размещается в одной клетке или на границе между двумя клетками.

Каждый простой робот умеет выполнять три команды-приказа, изменяющие положение робота: ВПЕРЕД, ПОВЕРНУТЬСЯ НАПРАВО, ПОВЕРНУТЬСЯ НАЛЕВО. Пиктограммы этих трех команд являются простейшими знаками-иконами (в смысле Пирса [4]), прямо изображающими выполняемое командой действие (см. рис. 1). Смысл этих команд воспринимается детьми в ходе демонстрации работы простых роботов и не требует специальных разъяснений взрослого или введения дополнительных терминов и понятий. Мгновенное усвоение семантики этих трех команд обеспечивается кинематическим опытом жизни ребенка в реальном мире.

Рис. 1. Пиктограммы команд, меняющих положение простого робота

Самый простой робот – *Ползун* – умеет выполнять только три команды-приказа, изображенные на рис. 1, и не имеет команд-вопросов. Таким образом, рассмотренные команды образуют минимальную структуру, обеспечивающую передвижение простого робота по игровому полю.

Простота описания всех возможных игровых обстановок и эффектов выполнения любых команд-приказов простых роботов вытекает из ***локальности*** правил построения обстановок и ***локальности*** правил выполнения команд. Локальность правил построения игровых обстановок состоит в том, что обстановку можно изменять в любой клетке произвольно, не согласуя эти изменения с состоянием соседних клеток. Локальность правил выполнения команд простого робота состоит в том, что любой простой робот занимает ровно одну клетку игрового поля, а изменения, которые он вносит в окружающую обстановку при выполнении любой команды-приказа, затрагивают только объекты, близкие к текущей позиции робота. Робот *Вертун* может закрасить клетку, в которой он стоит, но не может закрасить другие клетки, не меняя позицию. Робот *Тягун* может переместить только тот груз, который находится у него за спиной в соседней клетке. Робот *Двигун* может сдвинуть только один или два груза, которые расположены непосредственно перед ним. Аналогично ответы на команды–вопросы любого простого робота формируются локально и зависят только от состояния обстановки на игровом поле не далее нескольких клеток от текущей позиции робота.

С одной стороны, эта простота ускоряет освоение основных понятий программирования [6], делает освоение общих «правил игры по управлению роботами» доступным детям возраста 5+. На начальных этапах знакомства детей с программированием простота поведения роботов помогает усвоению основных понятий.

С другой стороны, подобно тому, как при освоении «игр с правилами» самостоятельный развивающий эффект дает не только сама игра, но и изучение и обсуждение правил, при изучении программирования, основанном на парадигме программного управления роботами, познавательное и речевое развитие детей может быть интенсифицировано за счет погружения ребенка в ситуацию, когда для исполнения роли программиста необходимо понять и освоить на практике заданные извне правила устройства игровых обстановок и правила работы роботов с достаточно сложным поведением. Эта ситуация моделирует практику работы профессионального программиста, собственно процессу создания программы всегда предшествует изучение инструментов, которые предоставлены для решения задачи.

Если в курсе «Алгоритмика для дошколят» основные интеллектуальные усилия прилагаются ребенком на этапе составления программ управления роботами с простым, интуитивно очевидным поведением, то при программировании роботов со сложным поведением, действующих в обстановках со сложной структурой, дополнительно требуются усилия другого рода, а именно усилия по осмыслению всех исходных данных задачи. Задачи освоения правил поведения роботов со сложным поведением и использования этих правил при индивидуальном и кооперативном составлении программ становятся, по нашему опыту, доступными детям возраста 6-7 лет, уже овладевшим азами алгоритмики, попа-

дают в зону ближайшего развития. И эту возможность не следует упускать.

Введение в курс алгоритмики для дошкольников роботов со сложным поведением позволит, в согласии с принципами действующего ФГОС дошкольного образования, «максимально разнообразить знания, умения и навыки ребенка». Ряд предметных навыков, приобретаемых детьми в ходе знакомства с роботами со сложным поведением, может быть отнесен к комбинаторике. В работе [7] обосновывается необходимость пропедевтической работы по развитию комбинаторных возможностей у детей дошкольного возраста. Согласно работам [7] и [8], особое внимание должно быть уделено знакомству детей с многофакторными зависимостями, поскольку «в обычной практике экспериментирование детей с многофакторными объектами носит эпизодический характер и поэтому не оказывает существенного влияния на их комбинаторное мышление». В работе [8], в частности, утверждается, что *осуществление реальных действий с предметами, своеобразный «материализованный этап» становления комбинаторного процесса, способствует знакомству детей с многофакторными зависимостями*. Индивидуальное и кооперативное составление старшими дошкольниками и младшеклассниками программ управления роботами со сложным нелокальным поведением оказывается целенаправленным практикумом, дающим возможность в наглядной форме ознакомиться с многофакторностью, необходимостью одновременного учета нескольких факторов.

В рамках годового курса «Алгоритмика для дошколят», состоящего из 30-35 занятий, изучение роботов со сложным поведением не оправдано. Положительный эффект от изучения роботов со сложным поведением можно получить на этапе углубленного изучения алгоритмов с обратной связью, то есть либо в курсе «Алгоритмика для первоклассников», либо на последнем году прохождения трехлетнего курса «Алгоритмика для дошкольников». Изучение роботов со сложным поведением дает развивающий эффект только при работе с детьми, достигшими возраста 6-7 лет и уже освоившими практически составление простейших алгоритмов с обратной связью.

2. Три отличия роботов со сложным поведением от простых роботов

Замечание. *Одним из возможных направлений усложнения поведения роботов могло бы быть добавление вероятностного поведения. В настоящей статье мы не рассматриваем такую возможность, ограничиваясь роботами с полностью детерминированным поведением.*

В настоящей статье описан робот *Паровозик* с достаточно сложным, геометрически нелокальным поведением в игровой среде, включающей пассивных роботов-партнеров «Паровозика», называемых *вагонами и цистернами*. Эти роботы-партнеры могут двигаться по игровому полю и менять свое состояние в результате воздействий *Паровозика*, но не имеют команд, доступных извне. *Паровозик* с партнерами – вагонами и цистернами – должен вводиться в курсе алгоритмики после изучения роботов *Двигун* и *Тягун*, которые, согласно легенде [5, Занятие 17], перемещают одиночные грузы – ящики и бочки – на грузовом космическом складе (см. рис. 2).

Паровозик можно рассматривать как усовершенствование *Тягуна*, а вагоны и цистерны – как усовершенствование ящиков и бочек. Ряд интуитивно понятных простых заданий по упорядочению ящиков и бочек на игровом поле *Двигуна* и *Тягуна* не может быть выполнен с помощью только Тягуна и требует привлечения к работе *Двигуна*. Однако многие похожие задачи по упорядочению вагонов и цистерн могут быть решены *Паровозиком* в одиночку. Алгоритмы при этом оказываются проще, и введение Паровозика в дополнение к простым роботам позволяет увеличить количество и сложность задач, которые могут быть решены детьми в отведенное программой курса время (см. рис. 3).

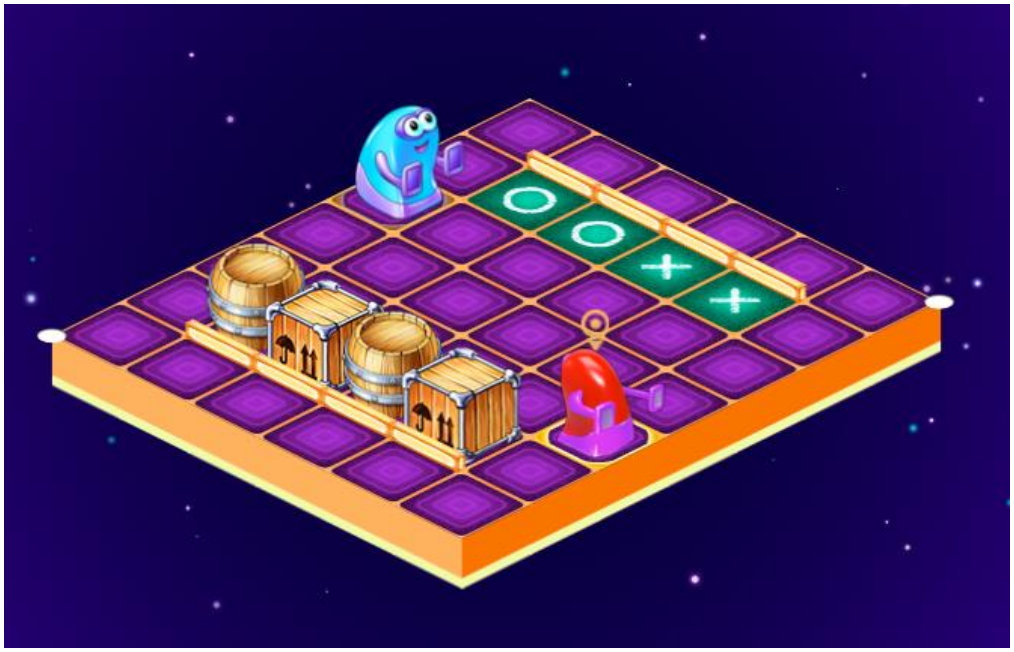


Рис. 2. *Двигун* и *Тягун* на космическом складе должны передвигать и перетаскивать ящики на базы, отмеченные крестиком, а бочки – на базы, отмеченные кружочком

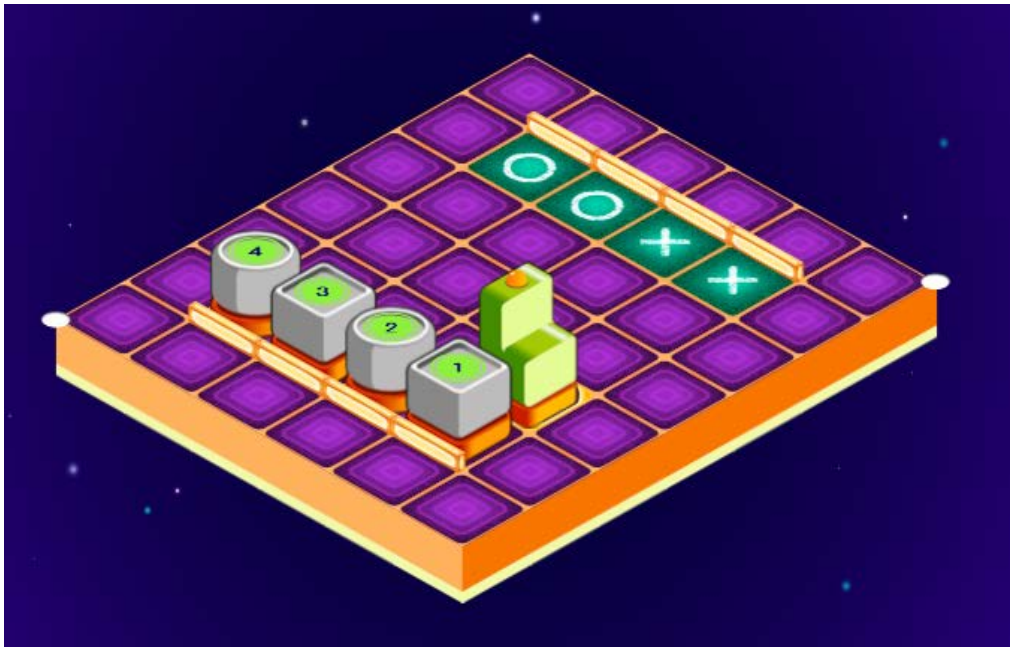


Рис. 3. *Паровозик* на космическом складе должен буксировать вагоны на базы, отмеченные крестиком, а цистерны – на базы, отмеченные кружочком

Паровозик работает на тех же космических складах, что и *Двигун* и *Тягун*, но перемещает по складу не ящики и бочки, а так называемые ***прицепы-вагоны*** и ***прицепы-цистерны***, а также ***составы*** – цепочки прицепов, которые Паровозик формирует, перемещаясь по складу и прицепляя или отцепляя в конце состава вагоны и цистерны с помощью магнитных ***сцепок***. Вагоны и цистерны обладают внутренними состояниями – они могут быть ***подсвечены*** и на каждом из них могут быть включены (активированы) одна или две магнитные сцепки. Эти внутренние состояния видимы – по изображению игровой обстановки можно понять, какие прицепы подсвечены и какие прицепы соединены сцепками друг с другом или с *Паровозиком* (см. рис. 5-7).

Правила выполнения команд *Паровозика* и структура игрового поля, на котором действует

Паровозик, в нескольких отношениях сложнее, чем команды и структура полей простых роботов.

2.1. Первое усложнение связано с продолжительностью выполнения одной команды. У простых роботов выполнение любой команды занимает одно и то же время, любая команда выполняется за один квант времени, за один ***такт***. У *Паровозика* есть команда ***буксировки*** (на один шаг), выполнение которой может занимать разное время в зависимости от конкретной игровой обстановки.

2.2. Второе усложнение связано с нелокальностью эффекта выполнения команд-приказов. Если у простых роботов эффект выполнения команды локализован в нескольких клетках, близких к позиции робота (одна-две клетки у *Вертуна* и до четырех клеток у *Двигуна* и *Тягуна*), то эффект выполнения одной команды *Паровозика* зависит от игровой обстановки и может проявиться: а) в клетках, далеких от позиции паровозика и б) не в одной клетке, а во многих, причем число таких клеток может быть сколь угодно большим и ограничено лишь общим числом клеток игровой обстановки. Задачи на нахождение этого числа в конкретных обстановках наглядны и имеют большой развивающий эффект.

2.3. Третье усложнение связано с нелокальностью информации, определяющей ответ *Паровозика* на команду-вопрос. В то время как ответ на команду-вопрос простого робота определяется состоянием нескольких клеток, близких к позиции робота, ответ на команду-вопрос *Паровозика* может зависеть от состояния большого количества клеток, в предельном случае – от состояния всех клеток игровой обстановки.

Понимание природы перечисленных трех усложнений доступно детям возраста 6-7 лет, и обсуждение этих усложнений дает заметный развивающий эффект. К необходимости понимания усложнения 1, переменной длительности выполнения команды буксировки, дети могут быть подведены в ходе выполнения заданий по кооперативному программированию, в которых участвуют два Паровозика. При выполнении таких заданий подсчет длительности выполнения команд буксировки часто оказывается необходимым для обеспечения синхронизации работы Паровозиков.

Замечание: с логической точки зрения составы являются *цепочками*, то есть примерами важной математической структуры, практическое освоение которой предусмотрено разделом 12.2. действующего ФГОС начального образования (извлечение из ФГОС НО [3] приведено в Приложении А). Ниже мы увидим, что составы разного типа описываются цепочками разного типа – ***однонаправленными*** или ***двунаправленными***. Выполнение заданий по программированию *Паровозика* способствует практическому освоению понятий «цепочка», «однонаправленная цепочка», «двунаправленная цепочка».

Игровое поле *Паровозика* состоит из ***клеток***. На поле могут находиться один или несколько *Паровозиков, каждый в своей клетке*, и несколько пассивных роботов-прицепов типа *вагон* и *цистерна*, которые могут соединяться с тем или иным *Паровозиком* или друг с другом с помощью ***сцепок***, образуя ***составы***. Между клетками могут располагаться стены, а в некоторых клетках могут быть установлены базы четырех типов: стоянка для вагона (крестик), стоянка для цистерны (кружочек), универсальная стоянка, способная принять и вагон, и цистерну (крестик в кружочке), база для заправки *Паровозика* (см. рис. 4).

Рис. 4. Изображение стен и различных типов баз на игровом поле Паровозика

Замечание. *Замечание. Свойства игрового поля Паровозика полностью определяются изображением игрового поля. Никакой скрытой информации, определяющей возможность выполнения определенной команды или сказывающейся на эффектах выполнения команды, правила работы Паровозика не вводят. В этом смысле правила выполнения команд Паровозика проще, чем правила ходов в шахматах, где возможность выполнения некоторых ходов не определяется по сложившейся на доске позиции, а дополнительно зависит от истории выполнения ходов в партии. Например, согласно шахматным правилам, рокировка короля и ладьи запрещена, если до этого момента король или ладья уже совершали ходы.*

Паровозик, вагоны и цистерны могут объединяться в **составы** с помощью магнитных сцепок. Сцепки включаются и выключаются по командам, которые Паровозик передает прицепам.

Составы бывают **активные**, возглавляемые Паровозиком, и **резервные**, которые могут быть возглавлены Паровозиком или присоединены в конец («в хвост») активного состава.

Сцепки позволяют Паровозику буксировать состав за собой как единое целое. Прицепы при этом перемещаются по складу не поворачиваясь, параллельно себе.

Порядок выполнения команды буксировки состава разберем на примере. На рисунке 5.а изображен состав, в который входят три вагона и две цистерны, соединенные пятью сцепками:

Паровозик ⇐ вагон 1 ⇐ вагон 2 ⇐ вагон 3 ⇐ цистерна 4 ⇐ цистерна 5.

Если дать Паровозику команду буксировки состава, то она будет выполняться в три этапа. Сначала Паровозик делает шаг вперед, освобождая клетку, в которой он находился. При этом магнитная сцепка «растягивается» и перемещается с левого борта Паровозика на задний борт и начинает тянуть вагон 1 на освободившееся место. Однако вагон 1 остается на месте до тех пор, пока Паровозик не освободит клетку полностью. В этот момент вагон 1 начинает двигаться и тянуть за собой вагон 2, а тот, в свою очередь, тянет за собой вагон 3. Таким образом, на втором этапе три стоящие в одну линию вагона начнут, как единое целое, перемещаться на клетку вперед, постепенно освобождая клетку, в которой находился вагон 3. Магнитная сцепка между вагоном 3 и цистерной 4 начнет растягиваться и тянуть за собой цистерну 4 на освободившееся место. Однако цистерна 4 остается на месте до тех пор, пока вагон 3 не освободит занимаемую им клетку полностью. В этот момент цистерна 4 начнет двигаться и потянет за собой цистерну 5. В результате на третьем этапе обе цистерны, как единое целое, перемещаются на клетку вперед, и по завершении третьего этапа весь состав переходит в положение, изображенное на рисунке 5.б.

Рис. 5.а. Состав из трех вагонов и двух цистерн, соединенных пятью сцепками.

Рис. 5.б. *Паровозик* сделал один шаг и в три этапа утянул состав за собой.

В составе, два положения которого изображены на рисунках 5.а и 5.б, *Паровозик* через сцепку подводит электричество к первому присоединенному к нему прицепу, тот подводит электричество к следующему и так далее. В результате электричество оказывается подведенным ко всем прицепам состава и все они становятся ***подсвеченными***.

Глядя на рисунки 5.а и 5.б, легко понять, что все изображенные на рисунке прицепы соединены в один состав, так как все прицепы подсвечены. Кроме того, факт вхождения прицепа в состав независимо подтверждается наличием на каждом прицепе одной или двух магнитных сцепок. А вот на рисунке 3 видно, что, хотя Паровозик и прицепы выстроены в одну линию, в состав они не соединены: прицепы не подсвечены и какие-либо сцепки на них отсутствуют. Такие прицепы, не включенные в какой-либо состав, называются ***одиночными.***

Проанализируем изображенный на рисунках 5.а и 5.б результат буксировки состава *Паровозиком* на один шаг вперед. На рисунке 5.б сцепки по-прежнему соединяют элементы состава в том же порядке, что и на рисунке 5.а, но они несколько изменили свое положение на элементах состава. Видно, что *Паровозик* по-прежнему сцеплен с вагоном 1 магнитной сцепкой, но эта сцепка «переехала» с левого борта *Паровозика* на задний борт, а у вагона 1 сцепка «переехала» с одной стороны вагона на другую соседнюю. Аналогично «переехала» в новое положение и сцепка между вагоном 3 и цистерной 4.

Конструкция *Паровозика* позволяет включить на нем только одну магнитную сцепку, которая может располагаться в одной из трех позиций: ***на заднем борту, на правом борту*** или ***на левом борту***. Конструкция любого прицепа позволяет включить (активировать) на нем не более

двух магнитных сцепок, каждая из которых может располагаться на любой из сторон, не занятых другой сцепкой.

Приблизившись к «одиночному» прицепу задней, правой или левой стороной, *Паровозик* может прицепить его к себе. При этом образуется состав с одним прицепом, у которого одна сторона соединена с Паровозиком сцепкой, а три стороны свободны. Такой прицеп называется ***хвостовым элементом состава***.

У хвостового элемента использована одна из двух возможных сцепок. По команде Паровозика вторая сцепка может быть добавлена к одному из трех свободных бортов хвостового прицепа и использована для присоединения к составу еще одного вагона или цистерны. После присоединения этот прицеп становится хвостовым (последним) элементом состава. Три свободных борта хвостового прицепа, по аналогии с бортами *Паровозика*, называются ***задний, правый*** и ***левый***. Например, если при движении состава из двух прицепов рядом с хвостовым вторым прицепом окажется вагон или цистерна, к этому хвостовому элементу по команде *Паровозика* можно присоединить третий прицеп, который станет новым хвостовым (см. рис. 6.а, 6.б).

Рис. 6.а. Состав из двух цистерн и одиночный вагон 3. Правый борт цистерны 2 граничит с одиночным вагоном 3.

Рис. 6.б. *Паровозик* присоединил вагон 3 к составу.

Присоединяя один за другим прицепы к концу состава, *Паровозик* может формировать составы произвольной длины. По команде БУКСИРОВАТЬ *Паровозик* с помощью гибких сцепок перемещает все прицепы состава. Состав, во главе которого есть *Паровозик*, называется ***активным***. Все элементы активного состава, включая *Паровозик*, подсвечены. Сам *Паровозик* считается активным составом, даже если к нему не присоединено ни одного прицепа. Хвостовым элементом такого состава считается сам Паровозик. Элементы активного состава упорядочены. Активный состав представляет собой ***однонаправленную цепочку.*** В такой цепочке есть начальный элемент и конечный элемент (мы назвали его хвостовым).

Начальным элементом всегда является *Паровозик*. У него есть возможность присоединения только одной сцепки. Если к *Паровозику* не присоединен никакой прицеп, то сам этот *Паровозик* считается активным составом, в котором первый элемент одновременно является последним, к которому с любой из трех сторон могут быть присоединены прицепы. Если присоединить к *Паровозику* один прицеп, получится активный состав из двух элементов: первый элемент – сам *Паровозик,* последний (хвостовой) элемент – прицеп (см. рис. 7).

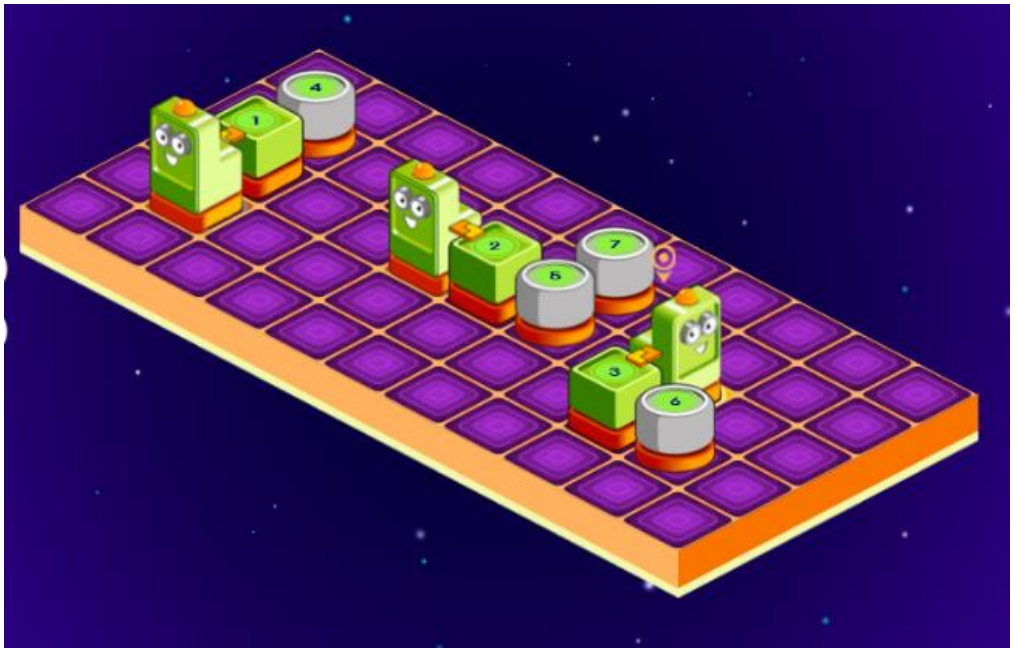


Рис. 7. Примеры составов с одним прицепом. Прицепы 4, 5, 6 и 7 одиночные, они ни в какие составы не включены

У хвостового элемента активного состава есть ровно одна вакансия на добавление еще одной сцепки. Эту вакансию можно использовать для присоединения еще одного прицепа к любому из трех бортов: заднему, правому или левому (см. рис. 8). Если присоединяемый прицеп одиночный, он становится хвостовым элементом удлиненного состава. Если присоединяемый прицеп – один из концевых элементов так называемого ***резервного*** состава (про него расскажем чуть позже), то присоединяется весь резервный состав, и в новом составе хвостовым элементом становится второй концевой элемент присоединяемого резервного состава.

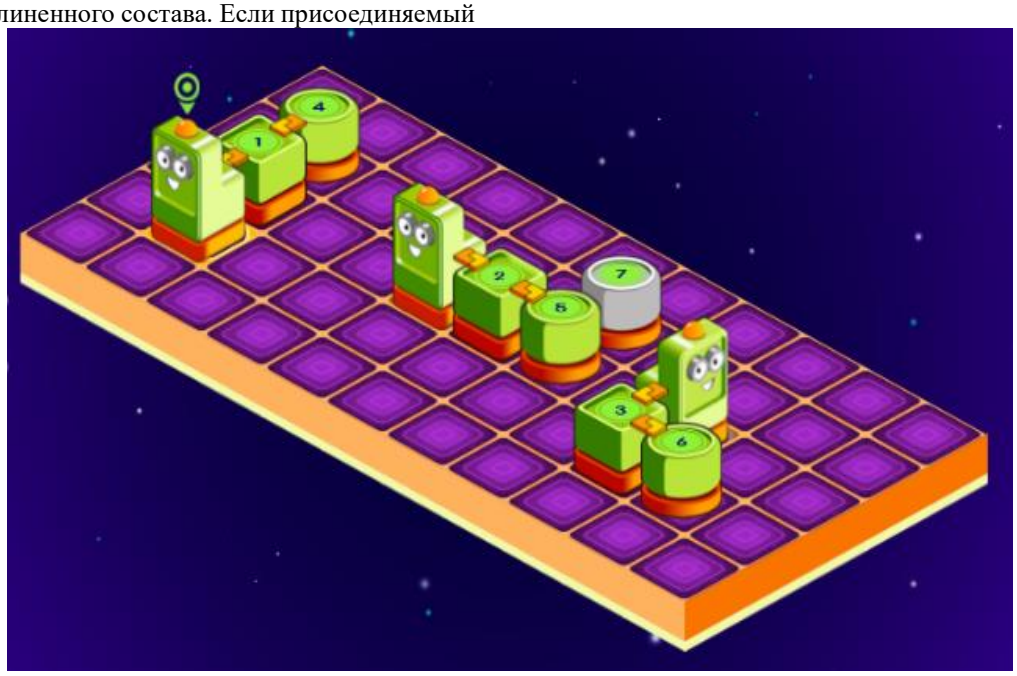


Рис. 8. Примеры составов с двумя прицепами. Цистерна 7 одиночная, ни в какой состав не входит, по команде *Паровозика* может быть присоединена к вагону 5

По инициативе *Паровозика* последний прицеп состава может быть отцеплен и превращен в одиночный прицеп. Подсветка этого прицепа окажется выключенной, а активный состав укоротится на один прицеп (см. рис. 9.а, 9.б).

Рис. 9.а. Состав из трех прицепов

Рис. 9.б. *Паровозик* отцепил вагон 3 от состава. Вагон 3 стал одиночным, его подсветка выключилась

Паровозик может тащить состав за собой, а может и разорвать (выключить) магнитную сцепку с составом, отойдя от состава на один шаг, как показано на рисунках 10.а и 10.б или повернувшись к составу «лицом», как показано на рисунках 10.а и 10.в. После разрывания сцепки состава с Паровозиком прицепы состава остаются сцепленными между собой, но подсветка всех прицепов выключается. Такой состав называется ***резервным***.

Рис. 10.а. Паровозик возглавляет состав из 5 прицепов

Рис 10.б. Разрывание связи по команде ВПЕРЕД

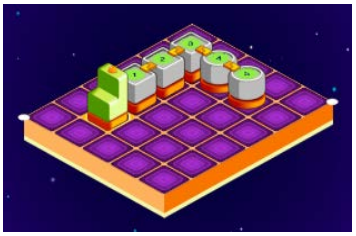


Рис. 10.в. Разрывание связи по команде НАЛЕВО

В резервном составе должно быть не менее двух прицепов, поэтому при разрывании связи *Паровозика* с составом из одного прицепа этот прицеп больше не называется составом, а называется ***одиночным*** прицепом. Резервный состав представляет собой ***двунаправленную цепочку***. В резервном составе есть два равноправных ***концевых*** прицепа, в каждом из которых задействована только одна сцепка и есть вакансия на включение второй сцепки для связи с *Паровозиком* или хвостовым элементом активного состава. Если *Паровозик* хочет возглавить резервный состав, то он может подойти к любому из двух концевых элементов этого резервного состава.

Например, на рисунке 11 в резервном составе из 6 вагонов концевыми будут вагон 1 и вагон 6, а в резервном составе из 6 цистерн концевыми будут цистерны 7 и 12.

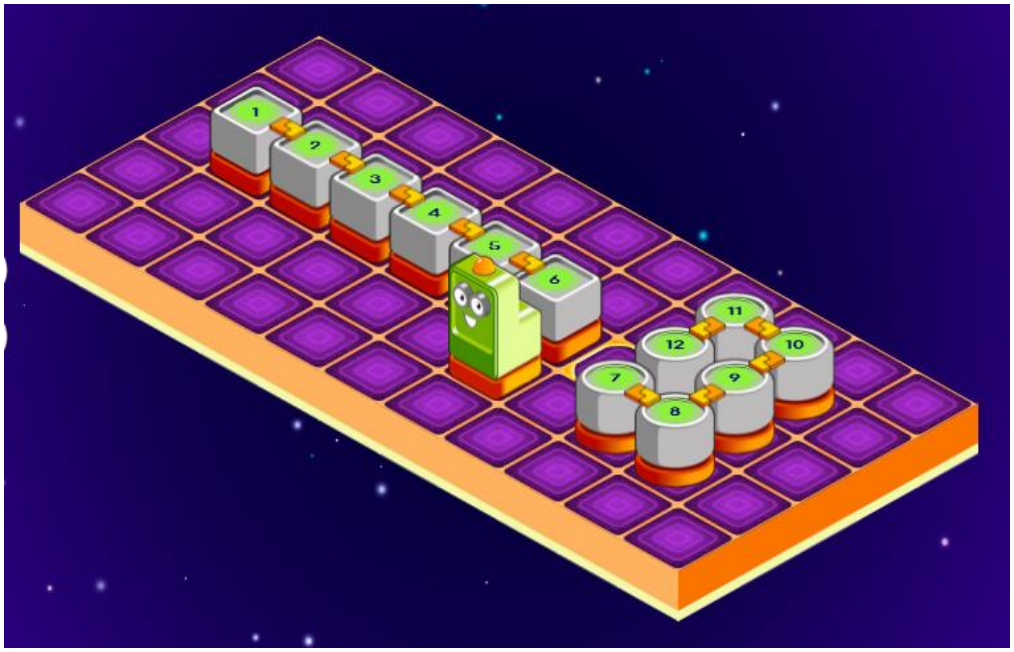


Рис. 11. На поле два резервных состава: из 6 вагонов и из 6 цистерн

Если *Паровозик* граничит задним, правым или левым бортом с концевым элементом резервного состава, то он может прицепиться к этому концевому элементу и тем самым возглавить резервный состав, превратив его в активный (см. рис. 12.а, 12.б).

Рис. 12.а. *Паровозик* граничит правым бортом с концевым элементом 6 резервного состава из 6 элементов.
Элементы состава не подсвечены.

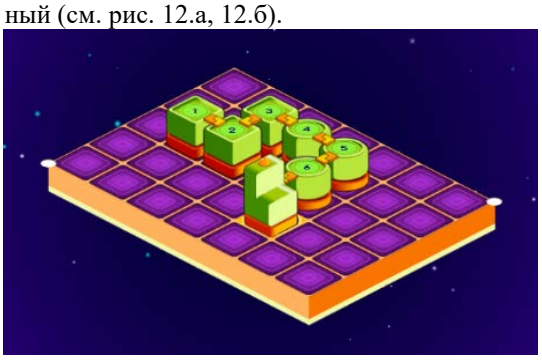
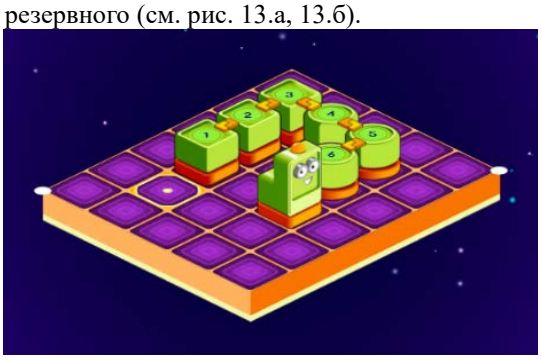


Рис. 12.б. *Паровозик* возглавил резервный состав, превратив его в активный.
Все элементы состава стали подсвеченными.

Кроме того, *Паровозик* во главе активного состава может «одним махом» присоединить к себе резервный состав, если хвостовой элемент активного состава граничит с концевым элементом резервного (см. рис. 13.а, 13.б).



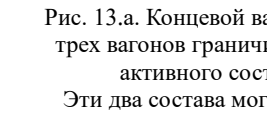


Рис. 13.а. Концевой вагон 3 резервного состава из трех вагонов граничит с хвостовой цистерной 4 активного состава из трех цистерн.
Эти два состава могут быть соединены в один

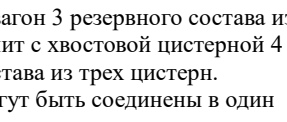


Рис. 13.б. По команде «прицепить» весь резервный состав присоединяется к концу активного состава и образуется активный состав из 6 прицепов.
На рисунке видно, что все 6 прицепов подсвечены

3. Перечень понятий, используемых при описании игрового поля Паровозика

Приведенное в разделах 2 и 3 описание игрового поля исполнителя Паровозик использует два десятка терминов:

обычная клетка, клетка с базой (стоянкой) для вагона, клетка с базой (стоянкой) для цистерны, клетка с универсальной базой, клетка с базой (заправкой) для Паровозика.

вагон, цистерна, прицеп, одиночный прицеп, подсвеченный прицеп, магнитная сцепка, состав, активный состав, хвостовой элемент активного состава, резервный состав, концевой элемент резервного состава, задний борт Паровозика или хвостового элемента, правый борт Паровозика или хвостового элемента, левый борт Паровозика или хвостового элемента.

Эта сложность вызвана существом дела: *Паровозик* обладает сложным поведением, и чтобы дать исчерпывающее описания этого сложного поведения, понятное человеку (или компьютеру), приходится вводить два десятка терминов. Заглянув в Приложение Б, можно увидеть, что описание поведения Паровозика, составленное для компьютера, столь же сложно, как и описание, составленное для человека.

Таким образом, количество понятий и терминов, связанных с исполнителем *Паровозик*, достаточно велико. И такой робот введен намеренно. Освоение понятий и терминов, необходимых для полного описания робота со сложным поведением, одновременно решает задачи ускорения и познавательного и речевого развития детей возраста 6+.

Во вводном курсе Алгоритмики дети прежде всего должны освоить правила работы *Паровозика* практически, научиться предсказывать результаты выполнения любых команд *Паровозика* в любых обстановках, научиться интуитивно использовать это понимание при составлении алгоритмов управления *Паровозиком*. Эта задача легко решается, если на практическое составление программ управления *Паровозиком* (индивидуальное и кооперативное) выделено достаточное время.

В.П. Зинченко писал [9]: «*Так или иначе, человек эффективно использует в поведении, деятельности, мышлении, созерцании построенную им картину мира. Иное дело, насколько он ее осознает и способен ли явить образ мира в слове, в картине, в действии, в поступке, в схеме, в формуле и т. д. Некоторым это удается, но даже в этом случае они не могут вразумительно рассказать, как они этого достигают. А. А. Ухтомский когда-то сказал, что люди сначала научаются ходить, а потом задумываются, как им это удалось*».

В терминах В.П. Зинченко практика программирования учит ребенка «*явить образ мира* ***в действии***» (командуя роботом с помощью пульта) или «*явить образ мира* ***в схеме и формуле***» (составляя программу будущего поведения робота), но не «***в слове***». Подтверждая мысль А.А. Ухтомского, ребенок может научиться управлять сложным роботом с пульта и составлять программы управления сложным роботом, успешно программировать робота, не задумываясь и не умея сказать словами, и даже не отдавая себе отчет о том, какие свойства робота он при этом использовал.

Получается, что задача практического усвоения правил поведения *Паровозика* и любого другого робота-исполнителя может быть решена ребенком без освоения и даже без введения приведенной выше терминологии, позволяющей описать поведение робота вербально. Поэтому

задача речевого развития детей на материале темы «программирование сложных роботов», должна быть сформулирована как самостоятельная и самоценная.

В разрабатываемых новых версиях курсах «Алгоритмика для дошкольников» и «Алгоритмика для первоклассников» задача ускорения речевого развития на предметном материале курса должна ставиться «отдельной строкой» и программы этих курсов должны предусматривать затраты времени на активное освоение детьми приведенного выше достаточно объемного списка терминов. Приобретение навыков практического составления и отладки программ решению этой задачи речевого развития помогает лишь частично. Должны быть разработаны методики освоения указанного набора терминов на бескомпьютерных этапах занятий.

4. Система команд робота Паровозик

У *Паровозика* 8 команд-приказов и 6 команд-вопросов (см. рис. 14.а, 14.б).

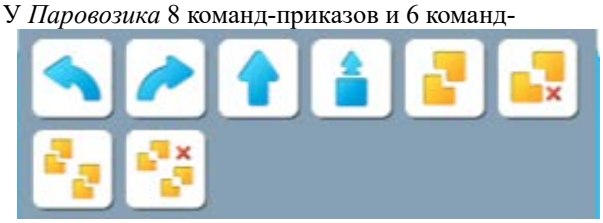


Рис. 14.а. Команды-приказы *Паровозика*

Рис. 14.б. Команды-вопросы *Паровозика*

Команды-приказы. Разберем 8 команд-приказов последовательно. Первые три команды-приказа – традиционные НАЛЕВО, НАПРАВО и ВПЕРЕД, их пиктограммы совпадают с пиктограммами аналогичных команд у простых роботов. Однако семантика этих команд у *Паровозика* сложнее. У простых роботов эти команды а) меняют положение самого робота и б) больше ничего не делают. Для *Паровозика* а) справедливо, а б) – нет. Если Паровозик возглавляет активный состав, выполнение команды ВПЕРЕД приводит не только к изменению положения самого *Паровозика*, но и к разрыву его связи с составом. В результате состав становится резервным, и подсветка всех его прицепов прекращается. Аналогично, если *Паровозик* был сцеплен с составом левым бортом и поступила команда НАЛЕВО, или был сцеплен правым бортом и поступила команда НАПРАВО, то *Паровозик* повернется к составу «лицом», связь с составом окажется разорванной и состав станет резервным.

Следующая команда БУКСИРОВАТЬ  приказывает *Паровозику* сделать шаг вперед, буксируя за собой весь состав, как было объяснено в разделе 3. Пиктограмма этой команды такая же, как у команды ТЯНУТЬ робота *Тягуна*. Если буксировать нечего, то есть Паровозик ни с чем не сцеплен, то при попытке выполнения этой команды *Паровозик* ломается.

Команда ПРИЦЕПИТЬ  – самая важная и самая трудная из команд Паровозика. Правила ее выполнения не очевидны и должны быть заучены наизусть. Дело в том, что есть несколько возможных вариантов выполнения этой команды, и нужно просто запомнить, какой из них был выбран создателями *Паровозика*.

Команда ПРИЦЕПИТЬ поступает *Паровозику*, но фактически выполняется хвостовым элементом активного состава, который паровозик возглавляет. В частном случае, когда к *Паровозику* ничего не прицеплено, этим последним элементом считается сам *Паровозик*. Напомним, что у хвостового элемента активного состава есть задний борт, правый борт и левый борт. При выполнении команды ПРИЦЕПИТЬ хвостовой элемент сначала исследует обстановку, граничащую с его задним бортом, потом обстановку, граничащую с его правым бортом и, наконец, с его левым бортом. Если в очередной рассматриваемой обстановке рядом с бортом расположена клетка и в этой клетке обнаруживается одиночный прицеп или концевой прицеп резервного состава, то этот прицеп соединяется с хвостовым элементом активного состава и на этом выполнение команды заканчивается. Если же ни в одной из трех рассмотренных обстановок нужный прицеп не обнаружен, *Паровозик* ломается.

Таким образом, после выполнения команды ПРИЦЕПИТЬ в хвост активного состава может быть добавлен одиночный прицеп или целый резервный состав.

Команда ОТЦЕПИТЬ  совсем простая. При ее выполнении хвостовой прицеп отцепляется от состава и становится одиночным. Разумеется, его подсветка выключается. Если к *Паровозику* не присоединен ни один прицеп, то при попытке выполнения команды ОТЦЕПИТЬ Паровозик ломается.

Оставшиеся две команды-приказа ПРИЦЕПИТЬ ВСЕ и ОТЦЕПИТЬ ВСЕ разберем после обсуждения команд-вопросов.

Команды-вопросы

Команды-вопросы ВПЕРЕДИ СВОБОДНО  и ВПЕРЕДИ ПРЕПЯТСТВИЕ  традиционные, такие же по смыслу, как у простых роботов.

Смысл остальных команд-вопросов очевиден из их названий

 МОЖНО ЧТО-ТО ПРИЦЕПИТЬ

 МОЖНО ЧТО-ТО ОТЦЕПИТЬ

 неверно, что МОЖНО ЧТО-ТО ПРИЦЕПИТЬ

 неверно, что МОЖНО ЧТО-ТО ОТЦЕПИТЬ

Наконец, команды-приказы Паровозика ПРИЦЕПИТЬ ВСЕ и ОТЦЕПИТЬ ВСЕ введены для оптимизации часто встречающихся при решении задач циклов:

	ПРИЦЕПИТЬ ВСЕ	**нц пока** МОЖНО ЧТО-ТО ПРИЦЕПИТЬ . ПРИЦЕПИТЬ **кц**
	ОТЦЕПИТЬ ВСЕ	**нц пока** МОЖНО ЧТО-ТО ОТЦЕПИТЬ . ОТЦЕПИТЬ **кц**

Эти команды выполняются за один такт, что вполне естественно, так как требуют только манипуляций с магнитными сцепками, которые, в отличие от механических манипуляций, выполняются почти мгновенно.

5. Примеры программ для робота *Паровозик*

5.1. Система команд *Паровозика* позволяет короткой программой перемещать большое количество объектов на большое расстояние

Дети с удовольствием составляют подобные программы, поскольку их результат зрим и нагляден. Рассмотрим задачу (см. рис. 15), в которой требуется переместить на большое расстояние большое количество объектов.

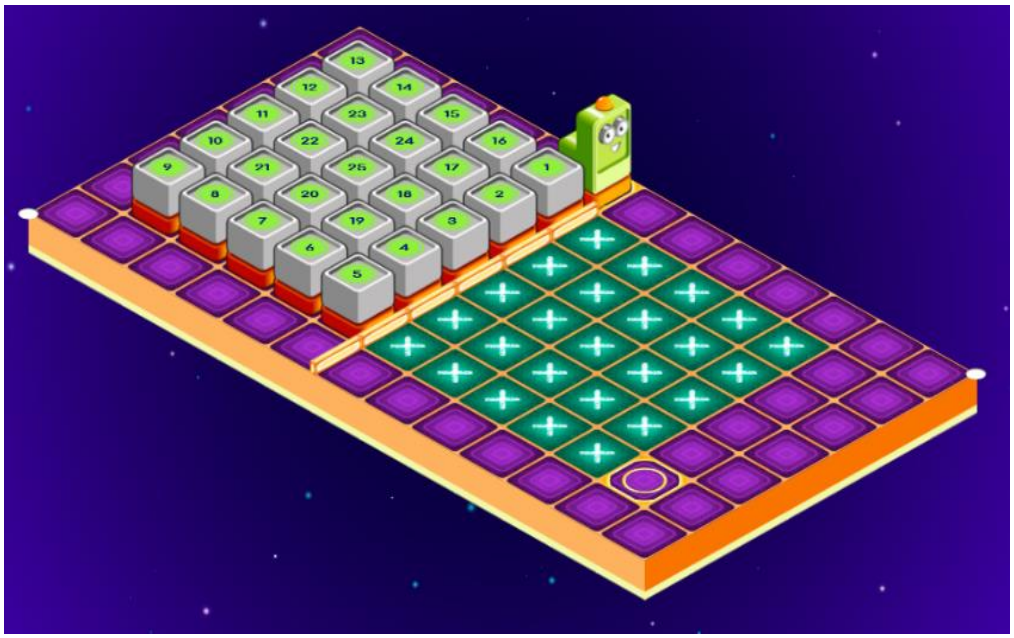


Рис. 15. Задача. Переместить 25 вагонов за стенку, в соседнее отделение склада

Составлять короткие программы для решения подобных задач удается за счет двух особенностей Паровозика.

Во-первых, *Паровозик* может одной командой ПРИЦЕПИТЬ ВСЕ соединить все вагоны в один состав (см. рис. 16):

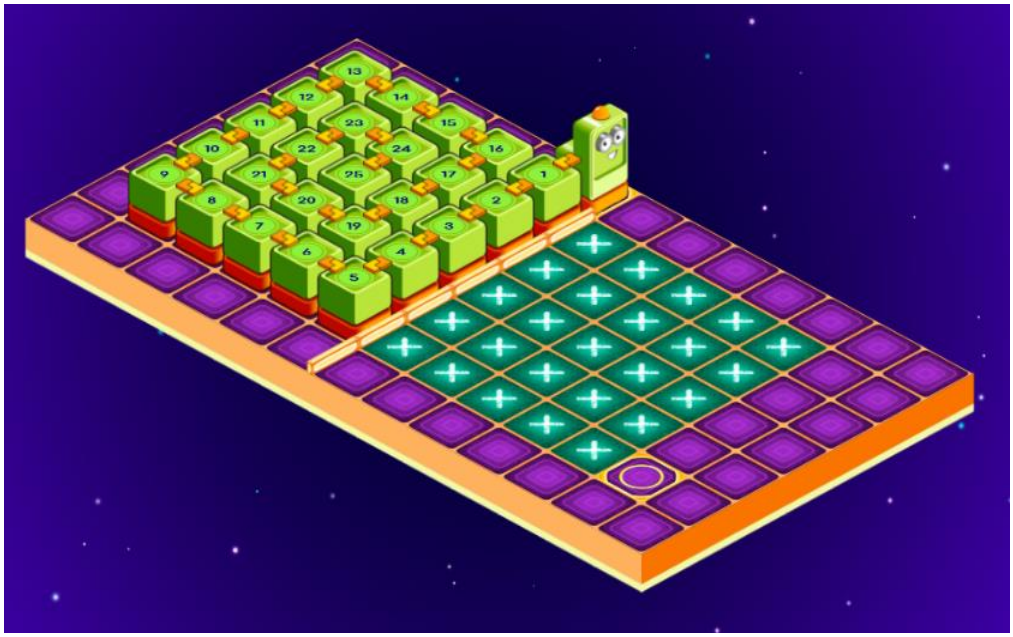


Рис. 16. Результат выполнения команды ПРИЦЕПИТЬ ВСЕ

В данной задаче команда ПРИЦЕПИТЬ ВСЕ эквивалентна выполнению за один такт 25 команд ПРИЦЕПИТЬ:

Первая команда ПРИЦЕПИТЬ не находит у заднего борта *Паровозика* прицепа, который можно было бы прицепить, но находит у правого борта Паровозика одиночный вагон 1 и прицепляет его к Паровозику. Вагон 1 становится хвостовым. С хвостовым вагоном 1 граничит вагон 16 по правому борту и вагон 2 по заднему борту. Вторая команда ПРИЦЕПИТЬ, в соответствии с правилами очередности поиска в этой команде, начинает поиск с заднего борта, находит у заднего борта хвостового вагона 1 одиночный вагон 2 и прицепляет его, вагон 2 становится хвостовым. Третья команда ПРИЦЕПИТЬ присоединяет к составу вагон номер 3 и т.д. В результате все вагоны с номерами от 1 до 25 оказываются по спирали присоединенными к составу.

Во-вторых, после того, как сформирован состав из 25 вагонов, каждая команда БУКСИРОВАТЬ за несколько тактов перемещает все эти 25 вагонов, поэтому перемещение всех вагонов на новое место (новое отделение склада) можно выполнить короткой программой (см. рис. 17), составление которой займет у ребенка не много времени:

Рис. 17. Перемещение 25 вагонов на большое расстояние может быть выполнено короткой обозримой программой

5.2. Система команд Паровозика позволяет короткой программой осуществлять сложные перестановки объектов

В задаче, показанной на рисунке 18.а, нужно переставить 4 объекта на новые места с изменением порядка. Стоянки, на которые *Паровозик* должен переместить вагоны и цистерны, выстроены в ряд:

кружочек ⇔ кружочек ⇔ крестик ⇔ крестик.

На первом этапе, выполнив фрагмент программы (см. рис.18.б), соберем все прицепы в один состав в порядке

Паровозик ⇐ вагон ⇐ вагон ⇐ цистерна ⇐ цистерна.

Рис. 18.а.

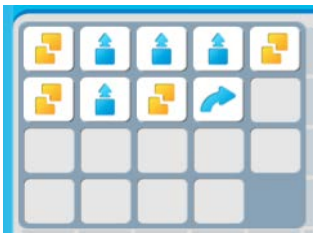


Рис. 18.б.

Рис. 18.в.

Рис. 18. Объекты собраны в один состав

Далее, запрограммируем проход Паровозика вдоль ряда стоянок от первого кружочка до последнего крестика. И когда Паровозик пройдет вдоль ряда и уйдет с этого крестика, все прицепы окажутся на нужных местах, и останется только расформировать состав командой «ОТЦЕПИТЬ ВСЕ». На рисунке 19 показано решение этой задачи.

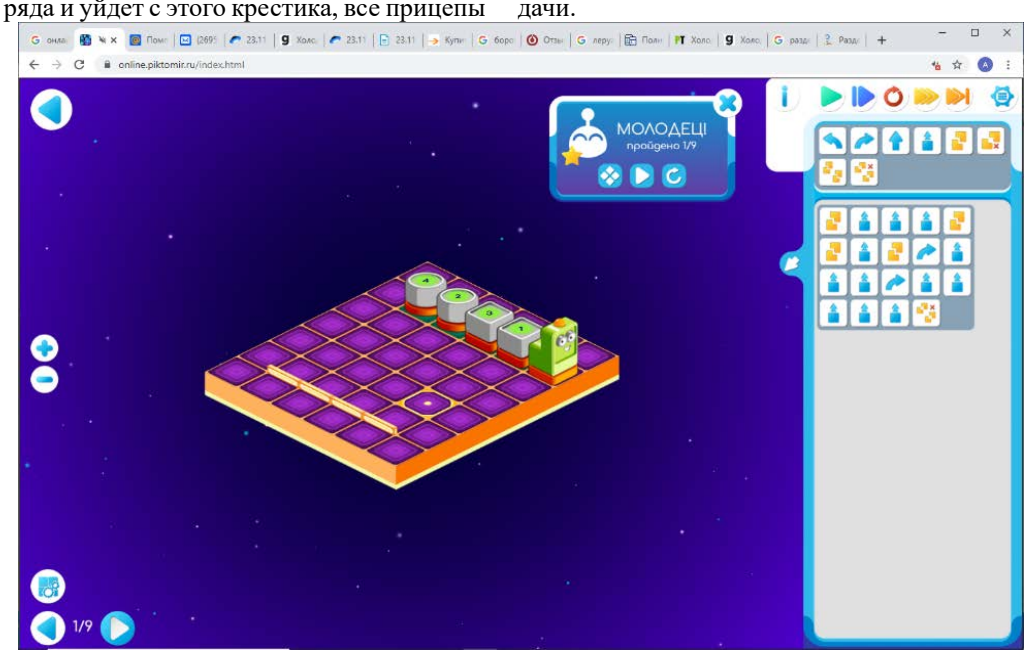


Рис 19. Состав перемещен на нужное место и расформирован

5.3. Система команд Паровозика позволяет с помощью двух роботов, работающих по одной программе, выполнять нетривиальные задания по разделению вагонов и цистерн

На рисунке 20 представлено задание, где с

помощью двух роботов надо разделить вагоны и цистерны и расставить их на свои места.

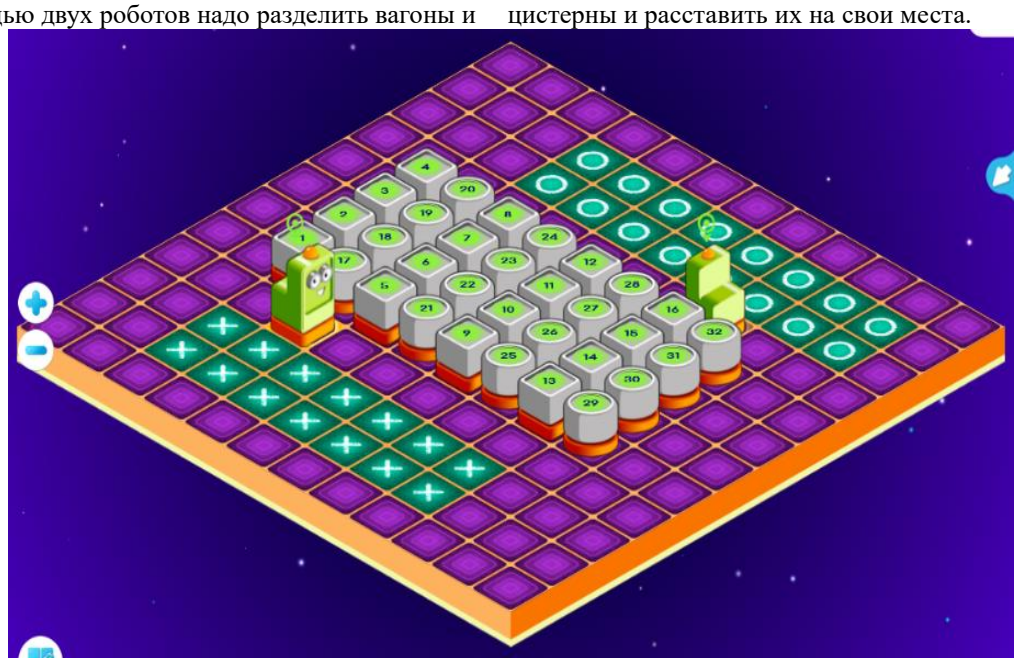


Рис. 20. Нетривиальное задание по разделению вагонов и цистерн

Программа решения этой задачи сначала может быть составлена без подпрограмм, методом проб и ошибок, с помощью так называемой *Копилки* ПиктоМира, в этой программе будет 59 команд. Затем, увидев в этой программе повторяющиеся последовательности команд, можно будет немного сократить программу, введя подпрограммы А и Б (см. рис. 21).

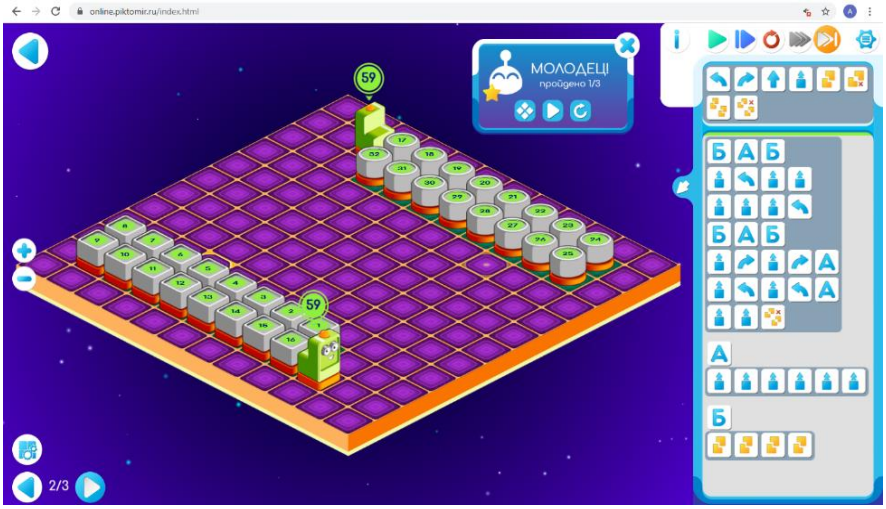


Рис. 21. Программа решения задания по разделению вагонов и цистерн

6. Заключение

Для робота *Паровозик* со сложным поведением удалось подготовить более сотни оригинальных заданий различного уровня сложности. Эти задания могут быть использованы в курсах программирования для разных категорий обучаемых, от дошкольников и младшеклассников до старшеклассников и студентов педагогических университетов.

Работа выполнена по теме 0065-2019-0010 «Разработка, реализация и внедрение семейства интегрированных многоязыковых сред программирования с автоматизированной проверкой заданий для учащихся образовательных организаций, ДОО, младшей, основной и старшей школы и студентов педагогических университетов» госзадания в отделе учебной информатики ФГУ ФНЦ НИИСИ РАН.

Приложение А

Извлечение из ФГОС начального образования

12. Предметные результаты освоения основной образовательной программы начального общего образования в части предметной области должны отражать:

… … …

12.2. Математика и информатика:

1) использование начальных математических знаний для описания и объяснения окружающих предметов, процессов, явлений, а также оценки их количественных и пространственных отношений;

2) овладение основами логического и алгоритмического мышления, пространственного воображения и математической речи, измерения, пересчета, прикидки и оценки, наглядного представления данных и процессов, **записи и выполнения алгоритмов**;

3) приобретение начального опыта применения математических знаний для решения учебно-познавательных и учебно-практических задач;

4) **умение** выполнять устно и письменно арифметические действия с числами и числовыми выражениями, решать текстовые задачи, **умение действовать в соответствии с алгоритмом и строить простейшие алгоритмы**, исследовать, распознавать и изображать геометрические фигуры, **работать с** таблицами, схемами, графиками и диаграммами, **цепочками**, совокупностями, представлять, анализировать и интерпретировать данные.

Приложение Б

Полуформальное описание реализации игрового поля и команд робота *Паровозик* в системе ПиктоМир

Далее Игровое поле *Паровозика* для краткости называется ***картой***. Любая карта имеет прямоугольную форму.

Раздел 1. Описания структуры карты и уровня *Паровозика*

1.1. Описания структуры данных элементов карты

У элементов карты есть 4 возможных типа – **Grass** (обычная клетка), **CarTarget** (клетка с базой для вагона), **BarrelTarget** (клетка с базой для цистерны), **CommonTarget** (клетка с универсальной базой). Каждый элемент карты имеет массив стенок вокруг него **walls**. Помимо этого, каждый элемент имеет данные **startForRobot**, которые хранят информацию о том, является ли элемент точкой старта для какого-либо робота, и если является – то для какого именно. Также элемент хранит два словаря **_robotsHere** и **_objectsHere**, которые хранят в себе **id** роботов и объектов, которые сейчас находятся на этой клетке (кроме пограничных моментов движения паровозика и объектов, каждый из словарей состоит из нуля или одного элемента). База (заправка) для паровозика представляет собой *задание о достижении конечной точки* и контролируется внешней системой, которая хранит данные о том, в какой клетке должен оказаться каждый робот, и рисует эти точки на карте; сами клетки этой информации не хранят.

1.2. Описание структуры данных карты в целом

Карта хранит в себе матрицу **mapElements** элементов карты, а также целые числа **width** и **height**, в которых хранятся ширина и высота карты. Помимо этого, карта хранит в себе массив **objects**, который содержит в себе данные обо всех объектах на карте.

1.3. Описание структуры данных уровня Паровозика

Уровень хранит в себе массив **robots**, который содержит в себе данные обо всех роботах уровня, массив **maps**, который хранит в себе данные обо всех картах уровня.

Раздел 2. Описания данных объектов на карте Паровозика

На карте есть объекты двух видов: **TrainCar** (вагон) и **TrainBarrel** (цистерна), оба являются наследниками класса **TrainObject** (прицеп) и отличаются только типом, который используется для определения корректности расстановки по базам, и внешним видом. Состав паровозика является двунаправленным списком. Для реализации этой структуры данных объект хранит в себе **parentObject** и **childObject** – данные о соседних прицепах этого прицепа в составе. Если у прицепа не нулевым является ровно один из этих объектов, то в случае, если состав активен, он является последним элементом активного состава, а если состав резервный – то его концевым элементом. По этим данным строится сам состав. Объект хранит в себе флаг **isMagnet**, который отвечает за то, является прицеп одиночным или нет. Также объект хранит **magnetRobotId** – номер робота, в состав которого он входит. Если его нет, то состав из таких прицепов является резервным. Иначе прицепы являются подсвеченными, состав является активным и относится к роботу с этим номером. Помимо этого, прицеп хранит **firstRobotId** – номер робота, для которого он является первым элементом состава, для корректной обработки кольцевых составов. Также прицеп хранит массив **connectionSprites** – массив магнитных сцепок прицепа, **position** – свою позицию на карте, **visualId** – номер, который отображается на прицепе.

Раздел 3. Описания данных Паровозика и его команд

Паровозик, как и прицеп, хранит массив **connectionSprites** – массив магнитных сцепок прицепа, position – свою позицию на карте, а также **direction** - свое направление и **id** – свой номер.

У робота есть основополагающая служебная функция **findTrain**(), которая находит доступный (активный или резервный) этому роботу состав и возвращает массив всех прицепов этого состава. Состав находится по следующим принципам: у его первого прицепа **firstRobotId** должен совпадать с **id** робота, у всех прицепов в составе **magnetRobotId** должен отсутствовать или быть равным **id** робота, **isMagnet** должен равняться **true**. Первый прицеп ищется с помощью **firstRobotId** или по правилам приоритета, остальные – с помощью **childObject** и **parentObject**.

Функция **move**(), отвечающая за движение робота, помимо передвижения самого робота очищает

firstRobotId у первого прицепа и **magnetRobotId** у всех прицепов состава – состав становится резервным.

Функция **carry**(), отвечающая за команду буксировки, помимо передвижения самого робота, получает активный состав этого робота, по кусочкам составляет анимацию передвижения робота и всех вагонов, перерисовывает все **connectionSprites** у робота и прицепов.

Функция **magnet()**, отвечающая за присоединение к составу новых элементов, находит текущий состав, берет его последний элемент и присоединяет к нему одиночный прицеп или другой резервный состав с помощью правильного прописывания всех свойств всех прицепов, участвующих в данной операции, чтобы в итоге получить корректный новый активный состав.

Функция **disMagnet()**, отвечающая за отсоединение от состава хвостового элемента, находит текущий состав, разрывает связь хвостового элемента с остальным составом, правильно прописывая все свойства хвостового элемента и полученного состава.

Остальные функции, такие как ***присоединить всё*** и ***отсоединить всё***, проверка условий, являются многократными вызовами уже описанных функций или их незначительными дополнениями.

The Need to Include Robotic Characters with Complex Behavior in a Textless Programming Course for Preschoolers and First Graders

A.G. Kushnirenko, K.A. Mashchenko, A.E. Orlovskii, N.A. Serebritskaia

Abstract. This article describes the arguments in favor of including in the final part of the textless programming course for preschoolers a workshop on compiling control algorithms for robots with complex behavior. An example of one such complex robot working in conjunction with passive partner robots is given, and it is shown that the introduction of such robots allows you to diversify and complicate the set of programming tasks available to preschoolers and first graders.

Keywords: Algorithm, robot, command executor, robot command repertoire, pictogram, programming, preschooler, first grader, introductory course, PiktoMir, complex behavior

Литература

1. Н.О. Бесшапошников, А.Г. Кушниренко, А.Г. Леонов, М.В. Райко, О.В. Собакинских. Цифровая образовательная среда пиктомир – опыт разработки и массового внедрения годового курса программирования для дошкольников. «Информатика и образование», 2020 (в печати).

2. А.Г. Кушниренко, А.Г. Леонов, М.В. Райко, Методические указания по проведению цикла занятий «Алгоритмика» для учащихся первого класса с использованием свободно распространяемой учебной среды ПиктоМир. [Электронный ресурс] // Свободно распространяемый методический материал на сайте ФГУ ФНЦ НИИСИ РАН. URL:https://www.niisi.ru/piktomir/Алгоритмика. 1 класс 19.09.2019. 1-14.pdf (дата обращения 23.11.2020).

3. Федеральные государственные образовательные стандарты. [Электронный ресурс], URL: https://fgos.ru/ (дата обращения 15.11.2020).

4. Семиотика [Электронный ресурс]// Википедия. Свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Семиотика (дата обращения: 23.11.2020).

5. А.Г. Кушниренко, А.Г. Леонов, М.В. Райко, Методические указания по проведению цикла занятий «Алгоритмика» в подготовительных группах дошкольных образовательных учреждений с использованием свободно распространяемой учебной среды ПиктоМир. [Электронный ресурс]// Свободно распространяемый методический материал на сайте ФГУ ФНЦ НИИСИ РАН. URL: https://www.niisi.ru/piktomir/Алгоритмика для дошкольников. 19.09.2019.pdf (дата обращения 23.11.2020).

6. В.Б. Бетелин, А.Г. Кушниренко, А.Г. Леонов. Основные понятия программирования в изложении для дошкольников. «Информатика и ее приложения», Т. 14 (2020), вып. 3, 56-62. DOI: 10.14357/19922264200308

7. Н.И. Хохлова, Принципы исследования комбинаторики у детей дошкольного возраста. «Вестник РГГУ. Педагогика. Психология». Вып. 1 (2012), 97-105, URL: https://vdocuments.mx/reader/full/-

5750a9b71a28abcf0cd25f4d (дата обращения 23.11.2020).

8. А.Н. Поддьяков. Развитие исследовательской инициативности в детском возрасте: диссертация на соискание ученой степени доктора психологических наук / Москва, 2001. 320 с.

9. В.П. Зинченко. Сознание и творческий акт. 2010 1 -588, Глава 12, стр. 419. // [Электронный ресурс], URL: https://publications.hse.ru/mirror/pubs/share/folder/aw7ywldro0/direct/82228022.pdf (дата обращения 23.11.2020).

Результаты освоения годовой программы «Алгоритмика для дошколят» подготовительными группами муниципального ДОУ

А.Г. Леонов[1], М.В. Райко[2], О.В. Собакинских[3], Н.В. Собянина[4]

[1]ФГУ ФНЦ НИИСИ РАН, г. Москва, Россия, dr.l@vip.niisi.ru

[2]ФГУ ФНЦ НИИСИ РАН, г. Москва, Россия, mila.rayko@gmail.com

[3]МБДОУ № 20 «Югорка», г. Сургут, Россия, natashaforever87@mail.ru

[4]МБДОУ № 20 «Югорка», г. Сургут, Россия, mixvel@yandex.ru

Аннотация. В статье приведены данные педагогического мониторинга курса «Алгоритмика для дошколят», проведенного в 2019-2020 учебном году в ДОУ г. Сургута, изложены результаты диагностики умений воспитанников по освоению основ программирования и проведен анализ эффективности усвоения материалов курса.

Ключевые слова: алгоритмика, информатика, ПиктоМир, программа, робот, ДОУ, ОВЗ

Годовая дополнительная общеобразовательная программа технической направленности «Алгоритмика для дошколят» с использованием свободно распространяемой цифровой образовательной среды ПиктоМир [1, 2] реализуется в МБДОУ № 20 «Югорка» г. Сургута для воспитанников старшего дошкольного возраста (6-7 лет) в течение нескольких последних лет. Курс дополнительного образования воспитанников разработан на основе методических рекомендаций по реализации курса «Алгоритмика», включает в себя 74 занятия, которые проходят дважды в неделю. На сайте дошкольного учреждения размещена утвержденная дополнительная общеобразовательная программа для детей 6-7 лет «Алгоритмика для дошколят» [3]. Аналогичные программы с сентября 2018 года реализуются во всех детских садах города Сургута [4] как отдельные курсы в рамках дополнительного образования, или как составляющая вариативной части основной образовательной программы дошкольного образования.

Необходимость анализа эффективности усвоения материалов курса воспитанниками обосновывает включение в программу такого важного компонента, как педагогический мониторинг результатов. Он осуществляется в два этапа: промежуточный - в декабре и итоговый мониторинг в апреле текущего учебного года. Программа не предполагает входящего мониторинга, что связано с отсутствием у детей на начало курса сведений о предметной области «алгоритмика» и знаний и умений относительно учебной среды ПиктоМир.

В докладе, опираясь на объективные результаты мониторинга 2019-2020 года, мы постараемся обосновать ряд выводов:

Педагогическое наблюдение, организованное, как на обычных занятиях, так и на диагностических, показывает желание детей заниматься алгоритмикой в течении года, высокую мотивацию и интерес к различным видам деятельности, которые им предлагаются:

- дети с радостью выполняют роли «робота» и «командира» в играх, предшествующих работе с реальным роботом;
- с удовольствием наблюдают за выполнением реальным роботом созданных педагогом и детьми программ;
- заинтересованы в создании корректных программ в течении всего курса, независимо от степени их сложности – этому во многом способствуют игровые приемы в подаче материала, отвечающие возрастным особенностям данного возраста.

Отсутствие особых сложностей в освоении материала курса, в овладении навыками составления простейших программ управления виртуальными и реальными роботами, что обусловлено грамотным, с постепенным усложнением, распределением материала курса от простейших программ к задачам более высокой сложности.

Полученные в процессе освоения курса навыки, относящиеся к категории метапредметных – такие, как создание алгоритмов действий, ориентировка на плоскости и в пространстве, планирование последовательности действий,

ориентировка на плоскости в определенной последовательности, умение устанавливать закономерности – демонстрируются детьми при решении задач из других областей, не относящихся к программированию.

Прослеживается настрой воспитанников на продолжение занятий программированием (алгоритмикой) в дальнейшем – в начальной школе.

Обратимся к более детальному анализу результатов диагностики, полученным в 2019-2020 году.

В программе принимали участие 50 воспитанников, 27 мальчиков и 23 девочки, из них 8 детей с ОВЗ.

Результаты мониторингов 2019-2020 учебного года показали, что 100% детей в целом справились со всеми предложенными заданиями. При этом около половины детей (25 из 50) и около половины детей с ОВЗ (4 из 8) справились со всеми заданиями без какой-либо посторонней помощи. Остальные дети выполнили большинство заданий самостоятельно, и небольшая подсказка со стороны педагога потребовалась при выполнении одного-двух заданий из десяти. Как правило, затруднения вызваны пропуском занятий. Небольшие сложности возникали у воспитанников при ограничении в некоторых заданиях доступа к отладочному средству «копилка команд».

Диагностика специальных умений воспитанников по освоению основ программирования показывает, что в итоге освоения курса дети:

- могут самостоятельно включить и выключить планшет;
- знают команды реального и экранных роботов и их обозначение в пиктограммах;
- способны составлять линейные программы;
- умеют составлять программы с использованием повторителей;
- могут составить программу с использованием одной и двух подпрограммы;
- способны найти ошибку и самостоятельно исправить ее.

На промежуточном этапе (в декабре) у ребенка уже есть опыт в составлении простейших программ достаточный для проведения мониторинга.

Опыт деятельности МБДОУ на протяжении более 5 лет по реализации программы дополнительного образования, а также успешное внедрение данного курса в дошкольных учреждениях города Сургута, которое курировалось нашим учреждением в качестве методической площадки, позволяет обосновать следующие аспекты:

Данный опыт реально воспроизвести в образовательных организациях для воспитанников 6-7 лет.

Обучение дошкольников старшего возраста основам программирования возможно осуществлять как на занятиях по дополнительному образованию технической направленности, так и в рамках вариативной части основных программ дошкольного образования.

Продолжительность курса может составлять от 2 до 9 месяцев, в зависимости от их частоты и насыщенности каждого занятия.

4. Для преподавания курса детям, педагогу необходимо иметь подготовку по направлению Алгоритмика для дошкольников.

5. Для проведения курса используется свободно распространяемый Учебно-Методический Комплект программного и методического обеспечения разработки ФГУ ФНЦ НИИСИ РАН [5, 6, 7, 8].

Состав комплекта:

а) бестекстовая учебная среда программирования «ПиктоМир»,

б) набор из 30 компьютерных игр по одной игре для каждого из 30 компьютерных занятий,

в) методическое пособие объемом 240 страниц.

Работа выполнена по теме 0065-2019-0010 «Разработка, реализация и внедрение семейства интегрированных многоязыковых сред программирования с автоматизированной проверкой заданий для учащихся образовательных организаций, ДОО, младшей, основной и старшей школы и студентов педагогических университетов» госзадания в отделе учебной информатики ФГУ ФНЦ НИИСИ РАН.

Приложение 1

Пример заданий декабрьского мониторинга

Для диагностики результатов обучения детям было предложено выполнить задания, предусмотренные в игре 9 «Олимпиада» программной среды «ПиктоМир», игра описана в методичке НИИСИ РАН на страницах 63-69. В отличие от регулярных занятий, где работа на планшетах продолжается около 15 минут, на данном занятии на работу на планшетах было отведено 30 минут с 5-минутным физкультурным перерывом. Автоматический сбор результатов выполнения заданий группой дошкольников не использовался. Вместо этого педагог подходил к каждому ребенку и видел на экране собранный системой ПиктоМир кумулятивный результат работы ребенка над заданиями.

Простое задание – уровень 1. В этом задании робот Вертун должен закрасить все клетки коридора, длина которого равна 6 (см. рис.1.а и 1.б).

Рис. 1.а. Шаблон программы задания с роботом Вертун

Рис. 1.б. Решение к заданию с роботом Вертун

Задание посложнее – уровень 7. В этом задании робот Двигун должен подвинуть мешающие ему ящики и перейти в назначенную клетку с зарядным устройством (клетка отмечена кружочком). Есть несколько вариантов решения, но только один из них позволяет уместить программу в заданный шаблон из 9 команд (см. рис 2.а и 2.б).

Рис. 2.а. Шаблон программы задания с роботом Двигун

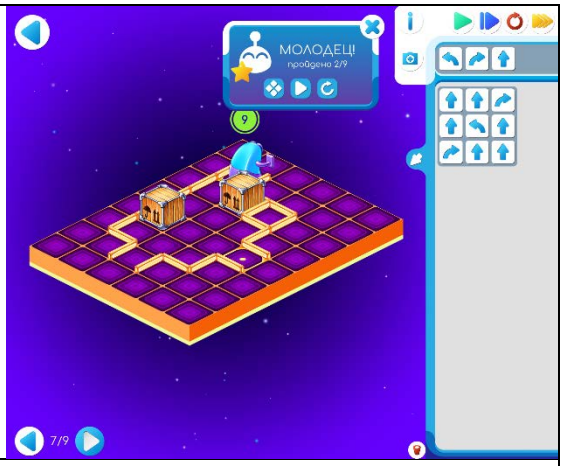


Рис. 2.б. Решение к заданию с роботом Двигун

Приложение 2

Пример задания весеннего мониторинга

Для диагностики результатов обучения детям было предложено выполнить задания, предусмотренные в мире «Базовый». Здесь робот Вертун должен закрасить букву «Г», причем шаблон предусматривает составление программы с одной подпрограммой (см. рис 3.а и 3.б).

Рис. 3.а. Шаблон программы задания с роботом Вертун. Буква «Г»

Рис. 3.б. Решение к заданию с роботом Вертун. Буква «Г»

Results of Mastering the Annual Program "Algorithmics for Preschooler" by Graduating Groups of Municipal Kindergartens

A.G. Leonov, M.V. Rayko, O.V. Sobakinskikh, N.V. Sobyanina

Abstract. The article presents the data of pedagogical monitoring of the course “Algorithmics for preschoolers” conducted in the 2019-2020 academic year at the preschool educational institution in the city of Surgut, presents the results of diagnosing the skillsof pupils in mastering the basics of programming and analyzes the effectiveness of mastering the course materials.

Keywords: algorithmics, informatics, PictoMir, program, robot, preschooler, disabilities

Литература

1. ПиктоМир – свободно распространяемая программная система для изучения азов программирования дошкольниками и младшими школьниками. https://www.niisi.ru/piktomir/ (дата обращения: 26.10.2020).

2. I.B. Rogozhkina, A.G. Kushnirenko. PiktoMir: Teaching Programming Concepts to Preschoolers with a New Tutorial Environment. “World Conference of Educational Technology and Researches”. – Procedia - Social and Behavioral Sciences, 28 (2011), 601 – 605. DOI: 10.1016/j.sbspro.2011.11.114

3. ДОПОЛНИТЕЛЬНАЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА технической направленности «Алгоритмика для дошколят», размещенная на сайте МБДОУ № 20 «Югорка» г. Сургута, http://ds20.detkin-club.ru/editor/21/files/Образование/допобразование/17cc9a2cc5a5c633f69013cf62835fe6.pdf (дата обращения: 26.10.2020).

4. А.Г. Кушниренко, А.Г. Леонов, М.В. Райко, О.В. Собакинских, Л.В. Шибаева. Влияние обучения программированию на основе системы "ПИКТОМИР" на развитие психологических новообразований старших дошкольников. «VIII Международная конференция «Воспитание и обучение детей младшего возраста» (ECCE 2019), России, Москва, 29 мая - 1 июня 2019 г. // Сборник «Воспитание и обучение детей младшего возраста». ISBN 978-5-19-011404-1

5. А.Д. Кисловская, А.Г. Кушниренко. Методика обучения алгоритмической грамоте дошкольников и младших школьников // Информационные технологии в обеспечении федеральных государственных образовательных стандартов: Материалы Международной научно-практической конференции, Россия, г. Елец, 16-17 июня 2014 г. Т. 2, 3-7. https://elibrary.ru/item.asp?id=22284368

6. Н.О. Бесшапошников, А.Г. Леонов. Пиктограммный язык программирования «Пикто» «Вестник кибернетики», Т. 28 (2017), № 4, 173-180.

7. А.Г. Куширенко, А.Г. Леонов, М.А. Ройтберг. Знакомим дошкольников и младших школьников с азами алгоритмики с помощью систем ПиктоМир и Кумир. «Труды НИИСИ РАН», Т. 5 (2017), № 1, 134–137.

8. Методические указания по проведению цикла занятий «Алгоритмика» в подготовительных группах дошкольных образовательных учреждений с использованием свободно распространяемой учебной среды ПиктоМир.

https://www.niisi.ru/piktomir/Алгоритмика%20для%20дошкольников.%2019.09.2019.pdf

Построение объектов дополненной реальности в динамически распознаваемой рукотворной среде

А.Г. Леонов[1], Н.О. Бесшапошников[2], М.С. Дьяченко[3], М.А. Матюшин[4]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, МГУ им. М.В.Ломоносова, Москва, Россия,

МПГУ, Москва, Россия, ГУУ, Москва, Россия, dr.l@vip.niisi.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, nbesshaposhnikov@vip.niisi.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, mdyachenko@niisi.ru;

[4]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, МГУ им. М.В.Ломоносова, Москва, Россия, matyushin@niisi.ru

Аннотация. Сегодня мало кто из экспертов берется предвосхитить потребности общества в тех или иных профессиях и компетенциях на 10-20 лет вперед. Современные технологии, такие как, дополненная реальность, виртуальная реальность, получают широкое распространение, в том числе, в образовании, как аппарат организации инновационного мультимедийного человеко-машинного интерфейса, когда объекты реального мира приобретают новые качества. Настоящие технологии, как правило позволяют лишь получать динамические изображения окружающей среды, дополненные смоделированными объектами. Актуальной задачей является распознавание рукотворных объектов с динамической интеграцией в виртуальном представлении в среду дополненной реальности, с возможностью использования на широко-распространенных моделях мобильных устройств. В данной статье рассматриваются методы построение объектов дополненной реальности в динамически распознаваемой рукотворной среде на примере цифровой образовательной среды ПиктоМир.

Ключевые слова: нейронные сети, дополненная реальность, роботы-исполнители, цифровая образовательная среда ПиктоМир

1. Введение

В пропедевтическом курсе алгоритмики в цифровой образовательной среде (ЦОС) ПиктоМир [1], в рамках которого дошкольники и младшие школьники осваивают основные понятия последовательного программирования [2], упор делается на составление программ для управления реальными и виртуальными роботами-исполнителями. При обучении программированию дошкольников и младших школьников ребенка учат не только составлять простейшие программки управления реальными роботами-игрушками или виртуальными роботами на экране планшета, но и формируют в голове у ребенка модель мира, в котором робот, программист, программа и компьютер тесно взаимодействуют между собой по понятным, четко установленным правилам [3]. На рисунке 1 представлено одно из заданий в образовательной среде ПиктоМир.

На начальном этапе этого освоения важнейшую роль играют материальные объекты -учебные пособия, такие как:

- игровое поле, составленное из разноцветных сочленяющихся квадратных мягких ковриков, размерами, достаточными, чтобы на нем мог стоять 6-летний ребенок;
- реальный робот-игрушка, перемещающийся по игровому полю повинуясь командам, подаваемым в форме слышимых детьми звуковых сигналов;
- кубики с нанесенными на их грани пиктограммами команд робота и точками, как на кубиках для настольных игр.

В курсе используются основанные на нейронных сетях программные модули:

- распознавание структуры программы управления роботом, составленной из кубиков по правилам языка «Пикто» [4];
- загрузка этой программы в учебную бестекстовую систему программирования «ПиктоМир» [5];
- распознавание комбинаторной структуры игрового поля из ковриков и загрузка этой структуры в ЦОС «ПиктоМир».

Рис. 1. Задание в ЦОС «ПиктоМир»

2. Использование распознавания рукотворных объектов и структур

В курсе ребенку ставится задача запоминания последовательности команд, которую он, ребенок, выдавал роботу для решения некоторой типовой задачи. Ребенок решает эту задачу с помощью кубиков, выкладывая их нужными пиктограммами вверх слева направо и(или) в несколько рядов один под другим в последовательности слева направо и сверху вниз (см. рис.2).

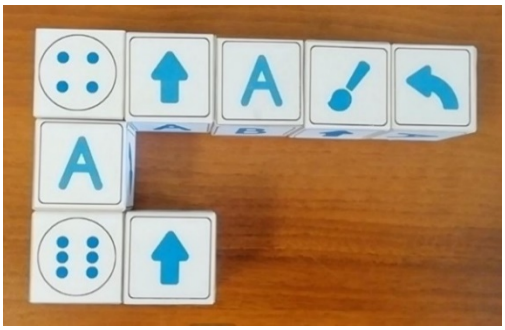


Рис. 2. Программа из кубиков

Ребенок «показывает» компьютеру (фотографирует) составленную из кубиков программу и видит, что компьютер пытается «понять» программу и показывает на экране, как он, компьютер, понял программу. Если ребенок подтверждает, что программа понята правильно, компьютер запоминает программу в своей памяти и готов выполнить эту программу по команде ребенка сколько угодно раз. Важно, что дети могут видеть, как компьютер берет на себя выполнение человеческих обязанностей. Поскольку ребенок сам умеет выполнять операции распознавания, запоминания и выполнения программ, выложенных из кубиков, вопрос о том, как именно компьютер выполняет эти операции, у детей не возникает.

Рис. 3. Игровое поле реального робота Ползуна

Позже в курсе начинает использоваться ЦОС «ПиктоМир» и освоенная детьми картина мира уточняется и усложняется. Например, дети осваивают, что некоторые длинные программы можно сократить, пользуясь кубиками-повторителями и кубиками-подпрограммами, и многое другое, включая программы достаточно сложной структуры с использованием циклов и вспомогательных алгоритмов.

На рисунке 3 представлено игровое поле реального робота Ползуна, которое повторяет поле виртуального Ползуна на экране планшета.

2.1 Распознавание программ из кубиков

Составление программы из материальных объектов используется в ряде робототехнических игрушек зарубежных и отечественных производителей (Робомышь [6], Primo Toys [7], Kibo [8, 9]). Однако, составляемые для этих игрушек программы не только слишком просты, но и, как правило, предлагают достаточно искусственную методику составления программы, например, предлагают использовать как элемент программы кубик с электрическим коннектором в нижней грани кубика, что позволяет однозначно «распознать» команду только если кубики размещаются на особом устройстве в специальных гнездах с электрическими контактами. Такая методика позволяет составлять только самые простые программы, напротив реальное распознавание изображений на свободно размещаемых ребенком на столе кубиках позволяет составлять программы достаточно сложной блочной структуры со вложенными управляющими конструкциями (см. рис.4).

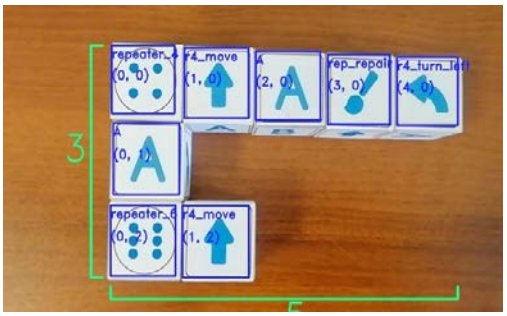


Рис. 4. Распознанная программа из кубиков

Разработка алгоритмов распознавания рукотворных объектов потребовала проведения работ по выбору архитектуры используемых нейронных сетей, разработки новых подходов к процедурам обучения сетей и совместного использования детерминированных и эвристических алгоритмов.

Задача генерации в памяти компьютера формального описания программы по фотографии определенным образом расположенных на плоскости кубиков, фактически сводится к несложной задаче по распознаванию на фотографии отдельных пиктограмм и более трудоемкому распознаванию двумерной структуры, которую образуют уже распознанные отдельные пиктограммы. Так, в примере реализации, встроенный в «ПиктоМир» программный модуль распознает на изображении отдельные пиктограммы и ряды пиктограмм и, в стиле языка «Питон», наделяет множество пиктограмм структурой программы на языке «Пикто», исходя из горизонтальных и вертикальных отступов между пиктограммами и рядами пиктограмм. Разработанные программные модули [10, 11], быстро и устойчиво решают эту задачу на современных широко-распространённых планшетах для изображений, полученных при ручной съемке выложенного ребенком множества из двух-трех десятков кубиков при всевозможных пространственных искажениях и вариациях освещенности.

2.2 Распознавание игрового поля

Следующая задача – распознавание комбинаторной структуры выложенного на полу игрового поля, по которому перемещаются роботы-игрушки. Так, например, в ЦОС «ПиктоМир», поле выложено из разноцветных сочленяемых квадратных ковриков с нанесенными на них цифрами. После того как дети под руководством преподавателя собрали из ковриков реальное игровое поле, и обсудили задание, которое должен выполнить робот, каждый ребенок «фотографирует» реальное поле с целью быстро получить его виртуальную копию в памяти своего планшета. Разработанный модуль распознает изображение с помощью нейронных сетей и восстанавливает комбинаторную структуру поля для последующей работы с полем средствами «ПиктоМира».

Распознавание игрового поля, являющегося прямоугольной плоской структурой, собранной из квадратных фрагментов - пазлов - одинакового размера, располагающихся друг к другу стык в стык, является комплексной задачей компьютерного зрения. Поскольку на игровом поле присутствует специфичная для игры разметка, моделируемая с помощью пазлов разного внешнего вида, необходима детекция не только и не столько границ поля, сколько распознавание его внутренней структуры.

В рамках решаемой задачи пазлы заведомо принадлежали к одной из двух категорий: пазлы с изображением одной цифры в диапазоне от 0 до 9; и одноцветные пазлы, каждый из которых монотонно окрашен в один из 3 цветов: зеленый, желтый или красный. Для успешного распознавания игрового поля необходимо получить координаты углов всех пазлов, тип каждого пазла (цифру, если пазл принадлежит первой категории, или цвет, если второй), и воссоздать заложенную во взаимном расположении пазлов табличную структуру. Таким образом, алгоритм распознавания игрового поля разбивается на 3 взаимозависимые стадии.

На первой стадии происходит детектирование пазлов первой категории. Поскольку изображение цифры имеет достаточно выраженную контрастную характерность, данная задача легко решается в терминах задачи детектирования объектов, для чего можно использовать предобученную сверточную нейронную сеть. В рамках данной задачи была использована нейронная сеть «ResNet-101 with Inception v2», предобученная на наборе данных COCO, которая затем дообучалась на собранных вручную и автоматически размеченных фотографиях пазлов первой категории.

После получения координат углов пазлов первой категории становится возможным осуществление второй стадии детектирования, а именно, детектирование пазлов второй категории. Поскольку пазлы второй категории имеют монотонную окраску, их контрастная характерность как отдельных объектов находится на низком уровне. По этой причине обычные сверточные нейронные сети плохо справляются с их детектированием. Однако, здесь применимы классические приемы компьютерного зрения, в частности, цветовые модели. С целью детектирования пазлов второй категории была обучена классификационная модель бустинга из 1000 деревьев глубины 3, разделяющая пространство цве-

тов RGB на жёлтую, зеленую и красную компоненты. Данная модель применяется на фотографии с закрытыми (так как их координаты уже получены на 1 стадии) пазлами первой категории, позволяя для каждого пикселя определить его принадлежность к одному из трех вышеперечисленных цветов. Для дальнейшего успешного разделения пазлов второй категории на игровое поле накладывается разумное требование-ограничение, заключающееся в том, что пазлам второй категории одного цвета запрещено соседствовать по общей стороне. При выполнении этого условия выделенные бустингом точки становится возможным кластеризовать, предварительно произведя с ними морфологическую обработку, а затем и классифицировать по преобладающему в кластере цвету. В рамках данной задачи в качестве морфологической предобработки использовалось гауссово размытие цветовых масок, на которых затем применялся алгоритм кластеризации DBSCAN с L1 метрикой. В ходе тестирования выяснилось, что описанная методика дает большое количество ложноположительных срабатываний цветовой модели за истинными границами игрового поля, в связи с чем было принято решение о добавлении в пайплайн обработки еще одного этапа – построение вспомогательной модели детекции ключевых точек. Описываемая модель предсказывает 4 угловые точки всего игрового поля, позволяя обрезать фотографию по ним перед применением цветовой модели. Тем самым исключаются упомянутые ложноположительные срабатывания. В качестве данной модели была выбрана 5- слойная сверточная нейронная сеть на базе блока Inception V2, предобученная на открытом наборе данных ImageNet, и доученная на вручную собранных и автоматически размеченных фотографиях игровых полей.

Третья и заключительная стадия распознавания заключается в восстановлении заложенной в игровом поле табличной структуры. В начале стадии 3 уже известны координаты углов всех пазлов, составляющих игровое поле, а также координаты угловых точек игрового поля. Восстановление табличной структуры фактически означает восстановление целочисленных координат пазлов в этой табличной структуре. С этой целью можно использовать одну из моделей для решения задачи табулирования [10]. Упомянутая задача включает в себя определение размера таблицы, а также непосредственное предсказание целочисленных табличных координат. В рамках задачи распознавания игрового поля использовалась unsupervised модель определения размера, основанная на анализе силуэта [13].

Рис. 5. Результат распознавания игрового поля

Суть ее состоит в том, что вдоль верхней и нижней стороны каждого пазла проводятся горизонтальные прямые. Затем к этим прямым применяется анализ силуэта с алгоритмом KMeans, который позволяет определить количество кластеров, что в данном случае соответствует высоте таблицы. Аналогично, проведя вертикальные прямые через боковые стороны, получаем ширину таблицы. Применяя теперь к координатам пазлов нейротабулятор [10], получаем их целочисленные табличные координаты. Выход данной модели составляет конечный результат в данной задаче, а ее применение завершает процесс распознавания игрового поля (см. рис.5).

На рисунке 6 представлена модель виртуального робота Ползуна.

Рис. 6. Модель виртуального робота Ползуна

3. Использование AR

Еще одна задача состоит в обогащении изображения реального игрового поля дополненной реальностью. Так в используемой в качестве примера ЦОС «ПитокМир», ребенок с помощью камеры планшета наблюдает на экране, в дополненной реальности, как виртуальный робот передвигается по реальным коврикам (см. рис. 7).

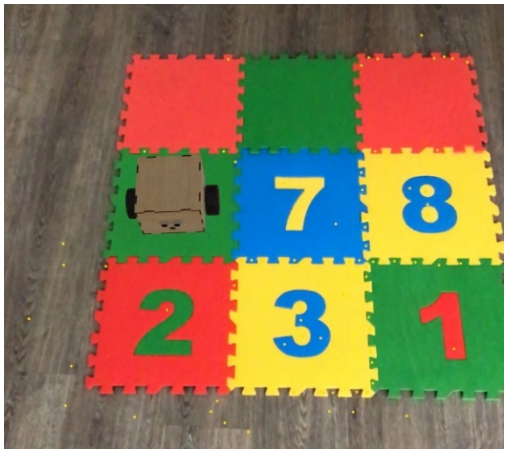


Рис. 7. Передвижение «дополненного» робота по реальным коврикам

Такое виртуальное учебное пособие привлекательно для педагога тем, что не требует подзарядки аккумулятора и не подвержено сбоям из-за дефектов сочленения реальных ковриков.

В примере реализовано два возможных режима работы с дополненной реальностью:

Тотальная виртуальность - и коврики и робот показываются как «дополненные».

Частичная виртуальность - - распознаются «реальные» коврики, а «дополненный» робот-Ползун движется по ним.

Первый режим работы не представляет собой никаких трудностей в реализации: достаточно использовать стандартный набор функционала современных мобильных библиотек дополненной реальности ARKit [13] для iOs и ARCore [14] для Android.

Работа во втором режиме напрямую зависит от распознавания игрового поля по плоской картинке с экрана устройства. После распознавания средствами библиотек дополненной реальности координаты углов ковриков на экране (т.е. на полученном изображении в момент начала распознавания) проецируются в виртуальное трехмерное пространство AR. Эта часть создает дополнительные сложности: так распознавание поля занимает существенное время (примерно 1-2 секунды), в то время как планшет может быть перемещен относительно изначальной позиции сьемки. Соответственно, координаты на распознанном изображении могут уже не соответствовать желаемым. Чтобы скорректировать передвижения устройства фиксируется проекция центра экрана в пространство AR в момент первоначальной сьемки и после распознавания. Затем вычисляется вектор сдвига в пространстве дополненной реальности и все спроецированные координаты углов сдвигаются на этот вектор.

Далее достаточно на основе углов вычислить центры клеток поля и передвигать между ними «дополненного» робота в соответствии с составленной программой.

Публикация выполнена при финансовой поддержке РФФИ научного проекта № 18-07-00901.

Construction of Augmented Reality Objects in a Dynamically Recognized Man-Made Environment

A. Leonov, N. Besshaposhnikov, M. Dyachenko, M. Matyushin

Abstract. Today, few experts undertake to anticipate the needs of society in certain professions and competencies for 10-20 years ahead. Modern technologies, such as augmented reality, virtual reality, are becoming widespread, including in education, as an apparatus for organizing an innovative multimedia human-machine interface, when objects of the real world acquire new qualities. These technologies, as a rule, only allow obtaining dynamic images of the environment, complemented by modeled objects. An urgent task is the recognition of man-made objects with dynamic integration in a virtual representation into an augmented reality environment, with the possibility of using it on most popular models of mobile devices. This article discusses methods for constructing augmented reality objects in a dynamically recognizable man-made environment using the example of the digital educational environment PiktoMir.

Keywords: neural networks, augmented reality, robots-executors, digital educational environment PiktoMir

Литература

1. I.B. Rogozhkina, A.G. Kushnirenko. PiktoMir: Teaching Programming Concepts to Preschoolers with a New Tutorial Environment. “World Conference of Educational Technology and Researches”. – Procedia - Social and Behavioral Sciences, 28 (2011), 601 – 605. DOI: 10.1016/j.sbspro.2011.11.114

2. А.Г. Кушниренко, А.Г. Леонов, М.А. Ройтберг. Знакомим дошкольников младших школьников азами алгоритмики помощью систем ПиктоМир и Кумир. «Труды НИИСИ», Т. 5 (2015), № 1, 134–137.

3. В.Б. Бетелин, А.Г. Кушниренко, А.Г. Леонов. Основные понятия программирования в изложении для дошкольников. «Информатика и ее применения», 14(3) (2020), 55–61. DOI: http://dx.doi.org/10.14357/19922264200308

4. А.Г. Леонов, Н.О. Бесшапошников. Пиктограммный язык программирования Пикто. «Вестник кибернетики», Т. 28 (2017), № 4, 173-180.

5. N. Besshaposhnikov, A. Kushnirenko, A. Leonov. Pictomir: how and why do we teach textless programming for preschoolers, first graders and students of pedagogical universities. ″Proceedings of the 13th Central & Eastern European Software Engineering Conference in Russia. – CEE-SECR '17″, New York, N.Y., United States, 2017.

6. Learning resources. [Электронный ресурс], URL: https://www.learningresources.com (дата обращения 15.11.2020).

7. Meet Cubetto. [Электронный ресурс], URL: https://www.primotoys.com (дата обращения 15.11.2020).

8. KIBO Robot Kits for Kids. [Электронный ресурс], URL: https://kinderlabrobotics.com/kibo/(дата обращения 15.11.2020).

9. Patent US20140297035A1 [Электронный ресурс], URL: https://patents.google.com/patent/US20140297035 (дата обращения 15.11.2020).

10. Н.О. Бесшапошников, А.Г. Леонов, М.А. Матюшин. Вопросы упорядочивания объектов на изображении с использованием нейросетевых и эвристических алгоритмов. «Вестник кибернетики», Т. 32 (2018), № 4, 136-142.

11. Н.О. Бесшапошников, А.Г. Леонов, М.А. Матюшин. О вариантах решения задачи распознавания табличной структуры по изображению в условиях отсутствия априорной информации. «Научная визуализация», Т. 12 (2020), № 4.

12. M. Chaudhary. Silhouette Analysis in K-means Clustering. [Электронный ресурс], URL: https://medium.com/@cmukesh8688/silhouette-analysis-in-k-means-clustering-cefa9a7ad111 (дата обращения 15.11.2020).

13. Introducing ARKit 4. [Электронный ресурс], URL: https://developer.apple.com/augmented-reality/arkit/ (дата обращения 15.11.2020).

14. ARCore overview. [Электронный ресурс], URL: https://developers.google.com/ar/discover (дата обращения 15.11.2020).

Подписано в печать 30.11.2020 г.
Формат 60x90/8
Печать цифровая. Печатных листов 25,75
Тираж 100 экз. Заказ № 1010

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