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

ТРУДЫ НИИСИ РАН

ТОМ 8 № 1

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2018

Тематика номера:
Информационные технологии, проектирование СБИС и наноэлектроника, математическое моделирование и визуализация, математические исследования и вопросы численного анализа,

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, вопросы численного анализа, история науки и техники

The topic of the issue:
Information technology, designing of VLSI and nanoelrctronics, mathematical modeling and visualization, mathematical researches and problems of numerical analysis.

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, problems of numerical analysis, history of science and of technique

Заведующий редакцией: Ю.Н.Штейников

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

СОДЕРЖАНИЕ

Методы снижения ресурсоемкости операционной системы жесткого реального времени

А.А. Асонов[1], А.Н. Годунов[2], В.А. Солдатов[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1]asonow@niisi.ras.ru, [2]nkag@niisi.ras.ru, [3]nkvalera@niisi.ras.ru

Аннотация: Несмотря на общий рост объемов доступной оперативной памяти, существует класс задач, работающих в жестком реальном времени, для которых оперативная память является лимитированным ресурсом. В задачах этого класса накладываются ограничения на ресурсы, используемые операционной системой реального времени (ОСРВ). Объем памяти, требуемой для ОСРВ, предлагается сократить за счет выполнения программного кода целевой системы из постоянного запоминающего устройства и путем расширения возможностей масштабирования уже существующей ОСРВ. Такой подход позволяет достичь существенной экономии памяти без больших переделок в ОСРВ, что позволяет снизить трудоемкость разработки и обеспечить высокую надежность ОСРВ. Некоторые части ОСРВ (обработка прерываний и исключительных ситуаций, поддержка многопоточности, служба времени, распределение памяти) являются обязательными, и их нельзя полностью исключить из целевого образа, а можно только сократить в той или иной степени. Все остальные части операционной системы могут быть либо существенно сокращены, либо вовсе исключены, если они не используются прикладной программой или пакетом поддержки модуля (ППМ). К ним относятся средства взаимодействия с сетью, файловые системы, блочные устройства, базовые операции ввода/вывода, средства отладки, условные переменные, очереди сообщений, таймеры, региональные настройки, программные каналы, сигналы, семафоры.

Ключевые слова: ОСРВ, обработка прерываний, обработка исключений, масштабирование, драйверы, конфигурирование, ОСРВ Багет.

Введение

В последние годы большое значение приобрела проблема импортозамещения элементной базы вычислительных систем [1]. В НИИСИ РАН были достигнуты значительные успехи по созданию отечественных микросхем. Так еще в 1997 году разработан первый в России 32-разрядный RISC процессор (1890ВМ1Т) со встроенным сопроцессором плавающей арифметики. Далее в 2008 году разработан первый в России 64-разрядный суперскалярный RISC процессор (1890ВМ5Ф). В 2011 г. закончена разработка комплекта микросхем с коммуникационной средой RapidIO. Каждая микросхема, представляющая собой систему на кристалле со встроенными сетевыми и коммуникационными каналами, может использоваться для создания супер-ЭВМ промышленного назначения.

Однако, несмотря на общий рост быстродействия и увеличение ресурсов, необходимых для проведения вычислительных работ, востребовано оборудование, для которого объем оперативной памяти является жестко лимитированным ресурсом [2-4]. На этом оборудовании решается класс управляющих задач, работающих в условиях жесткого реального времени, а общий размер оперативной памяти ограничен несколькими мегабайтами. Всю эту память условно можно разделить на две части. Одна из этих частей используется для прикладных программ, а другая ограничивает объем памяти, доступной для сервисов и драйверов операционной системы, под управлением которой выполняются прикладные программы пользовательской системы.

Кроме оперативной памяти имеется еще и постоянное запоминающее устройство (ПЗУ), которое используется для хранения неизменяемых данных.

Семейство операционных систем ОСРВ Багет 2.x [5,6], разработанное в ФГУ ФНЦ НИИСИ РАН, предназначено для создания управляющих систем жесткого реального времени. Первая версия ОСРВ Багет 2.x была выпущена еще в 2002 году. С тех пор состоялся выпуск еще 6 изданий ОСРВ этого семейства, которые используются более чем в 100 организациях. Все семейство ОСРВ Багет 2.x полностью удовлетворяет требованиям к информационной безопасности систем и прошло тестирование независимой системой тестов для операционных систем, разработанной в ИСП РАН [7]. Можно сказать, что ОСРВ Багет

2.x является распространенной и проверенной системой.

Для разработки прикладного программного обеспечения под управлением ОСРВ Багет 2.x используется комплекс, состоящий из двух ЭВМ, соединенных по сети. Первая из них называется инструментальной ЭВМ (компьютер, с операционной системой типа Linux), а вторая – целевой ЭВМ (ЭВМ, для которой разрабатывается программное обеспечение).

Разработка программного обеспечения ведется на инструментальной ЭВМ. Все функции и сервисы, используемые в прикладной системе, собираются редактором связей на инструментальной ЭВМ при создании целевого образа системы, выполняемой под управлением ОСРВ Багет 2.x. Целевой образ представляет собой один исполняемый файл формата bin или elf, который целиком загружается в оперативную память целевого процессора.

По своему функциональному назначению и предоставляемым сервисам, ОСРВ Багет 2.x полностью подходит для решения названного класса задач, однако, все существующие версии ОСРВ Багет 2.x, не удовлетворяют требованиям к объему оперативной памяти, который может быть доступен для ОС. Таким образом, ставится задача оценки существующих требований со стороны отдельных частей ОСРВ к используемым ресурсам и создания методики для управления объемом ресурсов, необходимых ОСРВ с целью построения компактной ОСРВ. Компактная ОСРВ должна выполнять все сервисы, требуемые пользователю, и помещаться в ограниченный объем оперативной памяти.

Для сокращения объема используемой ОСРВ оперативной памяти предлагается использовать:

– опцию компилятора –Os, которая позволяет строить объектные файлы минимального размера;

– размещение программных секций целевого образа в ПЗУ;

– масштабирование (scaling).

При создании целевого образа прикладной системы с помощью редактора связей можно настроить секции программного кода исполняемого файла на адреса, соответствующие ПЗУ. В этом случае программные секции не надо будет загружать в оперативную память. Скорость выполнения команд программы, считанных из ПЗУ, несколько ниже, чем команд, считанных из оперативной памяти, зато размер требуемой для системы оперативной памяти становится меньше на размер секций программного кода прикладной системы.

Масштабирование включает в себя как исключение какой-то части ОСРВ целиком, так и управление значением максимального количества объектов, которое обслуживается некоторой частью ОСРВ. Существующая ОСРВ Багет 2.x уже включает в себя возможность уменьшения целевого образа за счет масштабирования, однако получаемые результаты можно существенно улучшить путем переработки средств конфигурирования, инициализации и взаимодействия между отдельными частями ОСРВ. Поставленной задачей является разработка методов снижения ресурсоемкости операционной системы. Разработанные методы должны позволять создавать компактную ОСРВ, которая будет удовлетворять потребностям пользователя и занимать не более 1 МБ оперативной памяти. Образ, загружаемый на целевую ЭВМ, который включает в себя пользовательские программы совместно с сервисами ОСРВ и ППМ, должен помещаться в 2 МБ памяти.

С целью повышения мобильности ОСРВ из нее выделена отдельная часть, которая называется пакетом поддержки модуля (ППМ) и содержит ту часть ОС, которая зависит от конкретной аппаратуры (модуля). ППМ, в частности, содержит драйверы устройств и диспетчер прерываний (за исключением пролога и эпилога).

Целевой образ состоит из операционной системы, ППМ и прикладной программы. В компактных системах реального времени, которые предназначены для выполнения на оборудовании с ограничением на доступный объем оперативной памяти, граница между операционной системой и прикладной программой не так резко очерчена, как в случае традиционных операционных систем.

Прикладная программа выполняется в режиме ядра и виртуальная память не применяется. Это позволяет заметно сократить время обращения к функциям операционной системы, сократить время переключения контекста и увеличить скорость выполнения прикладных программ, сделав ее предсказуемой. Прикладная программа представляет собой совокупность потоков управления (пользовательских потоков управления).

Операционная система состоит из ядра и системных потоков управления. Ядро выполняет функции планирования, синхронизации и взаимодействия потоков управления, а также низкоуровневые операции ввода/вывода. Функции ядра выполняются в контексте вызвавшего его потока управления или функции обработки прерывания. Ядро представляет собой относительно небольшую часть ОС, функциями которой пользуются другие части ОС.

Системные потоки управления выполняют более сложные функции операционной системы, такие как ввод/вывод данных по сети или обмен данными с файловой системой. Использование системных потоков для сложных и протяженных во времени функций

ОС позволяет продолжать работу в параллель с выполнением этих функций. В рамках таких потоков выполняется часть функций, которые обычно выполняются в рамках обработчиков прерываний драйвера.

Как уже говорилось, в качестве основного способа сокращения объема ОСРВ предлагается масштабирование (scaling), однако не все части ОСРВ можно целиком исключить из целевого образа. Обязательными частями ОСРВ являются:

– обработка прерываний;

– обработка исключительных ситуаций;

– поддержка многопоточности (включая планирование потоков управления);

– служба времени;

– распределение памяти.

Некоторые из этих частей масштабируемы, за счет указания значения максимального количества объектов, которое обслуживается частью, но каждая из них не может быть полностью исключена из целевого образа. Все остальные части операционной системы дополнительно могут быть либо существенно уменьшены за счет исключения некоторой функциональности, либо вовсе исключены, если они не используются прикладной программой или ППМ. К этим частям относятся:

– взаимодействие с сетью;

– файловые системы;

– базовые операции ввода/вывода;

– средства отладки;

– очереди сообщений;

– библиотеки стандарта Си (включая математические функции);

– сигналы;

– средства синхронизации.

Рассмотрим последовательно каждую из перечисленных частей ОСРВ начиная с обязательных частей.

1. Обязательные части ОСРВ

Обработка прерываний и исключений. Обычно порядок выполнения машинных команд процессором строго определен. Команды выполняются последовательно одна за другой, исключениями являются команды передачи управления. Прерывания используются для того, чтобы изменить этот порядок – прервать выполнение текущей программы и начать выполнение другой.

Прерывания, в частности, могут происходить по запросу внешних устройств, например, по окончании операции ввода/вывода. Прерывания этого типа называются внешними, а также асинхронными, так как они возникают независимо от выполняемых процессором команд. Внешнее прерывание происходит, если прерывания не запрещены и имеется запрос на прерывание.

Прерывания происходят также при возникновении исключительных ситуаций при выполнении команд (например, если в команде указан неправильный код команды или указан несуществующий адрес). Некоторые команды предназначены для порождения прерываний (например, используемая отладчиком команда Break).

При возникновении прерывания управление попадает в диспетчер прерываний ОС. Диспетчер прерываний сохраняет состояние прерванной программы, выясняет причину прерывания и вызывает соответствующую функцию обработки прерываний.

Функции обработки исключительных ситуаций входят в состав ОС. Функции обработки внешних прерываний входят в состав ППМ (для тех устройств, которые поддерживает данный ППМ). Эти функции являются частью драйверов соответствующих устройств. Функция обработки прерывания по команде Break входит в состав отладчика. Перечисленные функции обработки прерываний регистрируются в таблице векторов прерываний при инициализации системы. Прикладная программа может установить свою функцию обработки прерываний для любого прерывания в любой момент времени.

Если при выполнении потока управления возникает исключительная ситуация, то операционная система в зависимости от конфигурации выполняет одно из следующих действий:

– посылает сигнал (в соответствии с POSIX);

– приостанавливает поток управления;

– выводит содержимое регистров и перезагружает систему.

Если исключительная ситуация возникает при выполнении функции обработки прерываний (например, в драйвере устройства), то по умолчанию система перезагружается.

Теоретически возможно построение операционной системы, в которой отсутствует механизм обработки прерываний, а все управление осуществляется по опросу. Однако реализация такого подхода вызовет полную переработку всех алгоритмов управления, применяемых в ОСРВ. Кроме того, существенным недостатком подобных систем является одинаковое время реакции на любые события, которые обрабатываются в системе, независимо от степени их важности. Также требуется, чтобы пользовательские программы, выполняемые под управлением такой системы, были написаны специальным образом, что вызывает существенные переработки уже существующего программного кода. И, наконец, так как исключительные ситуации невозможно замаскировать, то все-равно требуется механизм обработки этих событий.

В силу сказанного средства обработки прерываний и исключительных ситуаций

должны быть сохранены в ОСРВ. Возможность изменения реакции на исключительную ситуацию остается.

Поддержка многопоточночти. Для организации (псевдо-) параллельной обработки в современных ОС используются два понятия – процессы и потоки управления. Процессы являются держателями ресурсов (память, таблица открытых файлов и др.) и работают в значительной степени независимо друг от друга. Одной из функций ОС является защита процессов от нежелательного воздействия друг на друга. В рамках процесса может выполняться один или несколько потоков управления. Потоки управления, выполняемые в рамках одного процесса, совместно используют его ресурсы. Для систем реального времени, которые предназначены для выполнения на оборудовании с ограниченным объемом оперативной памяти, типичным является совместное использование ресурсов, поэтому мы будем рассматривать только потоки управления.

Команды программы одного потока выполняются до тех пор, пока не случится событие, обработка которого может привести к передаче управления другому потоку.

По сравнению с однопоточной системой многопоточная реализация позволяет достичь существенного упрощения программ в тех случаях, когда требуется обработка событий от нескольких устройств. Кроме того, повышается производительность за счет распараллеливания процессорных вычислений и операций ввода-вывода.

Поэтому часть ОСРВ, которая осуществляет функции порождения, управления и планирования потоками управления, должна оставаться в составе ОС. Для некоторой экономии памяти можно ограничить максимальное число потоков управления, которые могут выполняться параллельно.

Служба времени. Система всегда содержит, по крайней мере, одни часы с идентификатором CLOCK_REALTIME (системные часы). Значение этих часов интерпретируется как календарное время, то есть время (в секундах и наносекундах), истекшее с 0 часов 1 января 1970 года.

При наличии соответствующего оборудования в процессе конфигурирования системы могут быть созданы дополнительные часы. Их идентификаторы также определяются при конфигурировании системы (пакетом поддержки модуля).

Изменять характеристики аппаратных часов, например, изменять их разрешение, (программировать часы) можно в процессе инициализации системы, если это позволяет аппаратура.

Служба времени является важной составной частью любой ОС, а для ОСРВ служба времени является одной из важнейших, так как с ее помощью ведется отсчет текущего времени, измерение и задание временных интервалов, быстрое выполнение системных действий с гарантированной верхней границей продолжительности. В силу сказанного эта часть ОСРВ Багет 2.x не может быть исключена из состава ОС.

Распределение памяти. Рассмотрим ОСРВ с функциями (динамического) распределения памяти в соответствии со стандартом С, а также с механизмом распределения памяти на основе так называемых пулов памяти.

К стандартным средствам относятся функции:

– calloc() – выделение и очистка памяти;

– free() – освобождение памяти;

– malloc() – выделение памяти;

– realloc() – перераспределение памяти.

Использование перечисленных функций не требует знакомства с пулами памяти (пулы памяти используются неявно). Однако при порождении потока управления можно указать, какой пул памяти будут использовать эти функции в контексте порождаемого потока (см. ниже).

Пул памяти может быть создан функцией memPoolCreate() на свободном участке памяти. Непосредственно после создания пула вся его память (за исключением небольшого участка, занятого управляющей информацией) объявляется свободной и может быть выделена для прикладной программы или ОС. В дальнейшем к пулу памяти могут быть добавлены другие, вообще говоря, несмежные участки памяти функцией memPoolAdd(). Таким образом, пул памяти может содержать один или несколько, вообще говоря, несмежных участков памяти.

Память, находящаяся в пуле, используется следующим образом. В начале нужно получить участок памяти из пула с помощью функции memAlloc(). После получения участка памяти он может использоваться прикладной программой. Когда необходимость в данном участке памяти отпадет, его следует освободить (вернуть в пул памяти) функцией memFree(). При инициализации системы вся свободная память объединяется в системный пул.

У каждого потока управления имеется пул памяти, который используется по умолчанию. Этот пул памяти используется функциям стандарта С calloc(), free(), malloc(), realloc(), а также функциями memAlloc(), memFree(), если пул памяти не указан (указано NULL). Из этого же пула памяти берется память под стек потока управления (если не указан атрибут stackaddr). Используемый по умолчанию пул памяти указывается при порождении потока управления (атрибут mempool). Если при порождении потока не указать пул памяти, то будет использоваться системный пул. Использование пулов памяти позволяет избежать фрагментации памяти, а также позволяет

уменьшить нежелательное влияние одних потоков на переменные других потоков, возникающее при наличии ошибок в программном коде.

Организация и управление памятью вычислительной машины является одним из важнейших факторов, определяющих построение ОС, следовательно эта часть ОСРВ Багет 2.x не может быть исключена из состава ОС.

2. Необязательные части ОСРВ

К необязательным частям ОСРВ относятся те, которые могут быть полностью или частично исключены в случае, когда они не используются прикладной программой или ППМ. Чем больше ресурсов требуется для исключаемой части ОСРВ, тем больший выигрыш будет получен при исключении части из целевого образа.

Взаимодействие с сетью. Сетевое программное обеспечение в ОСРВ обычно базируется на использовании семейства протоколов TCP/IP, применяемого для передачи данных в сети Интернет. Важная особенность семейства протоколов TCP/IP – обеспечение межсетевого взаимодействия. Они позволяют интегрировать разнородные сети, содержащие различные компьютеры, работающие под управлением различных операционных систем. Для передачи данных из одной сети в другую используются шлюзы (gateways) – компьютеры, подключенные одновременно к двум или большему числу сетей. Другими словами, протоколы семейства TCP/IP дают возможность объединить несколько различных (локальных) сетей в одну общую сеть.

Подключение сетевых средств возможно только при условии подключения средств поддержки таймеров и очередей сообщений.

Конфигурирование сети позволяет включить (или не включать) сетевые средства в целевой образ системы. Если сетевые средства включаются в образ, то уменьшить их объем можно за счет исключения из образа части поддерживаемых протоколов (RPC, SLIP, PPP), сетевых утилит (ping, telnet-клиент, telnet-сервер, получения времени по сети), таблицы компьютеров, количества сетевых буферов.

Файловые системы. Операционная система может обеспечивать доступ к одной или нескольким файловым системам. Каждая файловая система содержит файлы и каталоги, образующие древовидную структуру. Корневая файловая система присутствует всегда. Доступ к другим файловым системам обеспечивается при условии их установки (монтирования).

Корневая файловая система содержит корневой каталог, а также каталог /dev, который содержит все специальные файлы. При монтировании корневой каталог монтируемой файловой системы помещается в корневой каталог корневой файловой системы (иначе говоря, точка монтирования всегда находится в корневом каталоге).

Операционная система поддерживает локальные файловые системы VFAT, ISO9660 (для компакт-дисков) и TAR, а также удаленные файловые системы NFS и FTP. Поддержка файловых систем требует включения блочных устройств. Поддержка удаленных файловых систем требует включения сетевых средств. Для удаленной системы NFS можно указать поддержку NFS-клиента и можно дополнительно установить поддержку NFS-сервера. Для файловой системы FTP поддерживается только FTP-клиент.

При конфигурировании операционной системы можно указать типы файловых систем, которые нужно поддерживать, и те файловые системы, которые надо автоматически смонтировать. Эти файловые системы будут установлены при инициализации целевой системы.

Какие именно файловые системы будут использоваться, указывается при конфигурировании целевого образа. Также при конфигурировании определяется максимальное количество файлов, которые одновременно могут быть открытыми. Можно полностью исключить поддержку файловых систем.

Блочные устройства. При конфигурировании можно ограничить количество буферов и размер буферной памяти, которые отводятся под блочные устройства.

При отключении средств поддержки блочных устройств исчезает возможность работы с такими файловыми системами, как VFAT, ISO9660, NFS и FTP. Корневая файловая система, каталог /dev и файловая система TAR могут быть включены в образ и при исключении поддержки блочных устройств.

Средства асинхронного ввода/вывода. Стандартные функции ввода/вывода read() и write() приостанавливают выполнение потока управления до завершения операции. Асинхронные операции ввода/вывода позволяют продолжить выполнение потока управления параллельно с операцией ввода/вывода.

Асинхронные операции чтения (aio_read()) или записи (aio_write()) устанавливают в очередь заявку на выполнение операции чтения (соответственно, записи) и, если возможно, запускает операцию и, не дожидаясь ее завершения, возвращают управление запустившему их потоку.

Средства асинхронного ввода/вывода не конфигурируются, т. к. все функции, относящиеся к этой части ОСРВ, попадают в целевой образ только в случае их вызова из прикладных программ пользователя.

Потоки ввода/вывода. Язык программирования Си поддерживает множество функций стандартных библиотек для потокового ввода и вывода. Эти функции составляют основу заголовочного файла стандартной библиотеки языка Си <stdio.h>.

Функциональность ввода-вывода языка Си по текущим стандартам реализуется на низком уровне. Язык Си абстрагирует все файловые операции, превращая их в операции с потоками байтов, которые могут быть как «потоками ввода», так и «потоками вывода».

В ОСРВ можно предложить два способа отключения функций поддержки работы с потоками ввода/вывода:

– отключение всех функций работы с потоками (FILE *);

– отключение только функций printf(), fprintf(), vprintf(), vfprintf().

Исключаемые функции заменяются на заглушки, которые возвращают ENOSYS.

Форматированный ввод/вывод. В ОСРВ Багет 2.x используется три функции для форматированного вывода:

– kprint();

– syslog();

– printf().

Функция printf() соответствует стандартам POSIX и Си. Она использует функции форматных преобразований, а также функции обслуживания потоков ввода и вывода (FILE *). Собственно вывод отформатированных сообщений осуществляется функцией write(), которая соответствует стандарту POSIX. Сообщения выводятся в файл (в том числе на терминал). Функцию printf() нельзя использовать в обработчиках прерываний.

Функция syslog() соответствует стандарту POSIX. Для форматирования она использует функцию vsnprintf() и в силу этого в образ ОС попадают функции форматирования, а также функции обслуживания потоков ввода и вывода (FILE *). Отформатированное сообщение помещается в очередь, и впоследствии выводится специальным потоком (syslog demon). Возможен вывод в файл (в том числе на терминал) и/или в память (во флэш-память и в обычную память). Последние сообщения (помещенные в очередь, но еще не выведенные) могут быть потеряны в случае краха системы. Возможен вывод, минуя очередь (обычно используется в случае тяжелых ошибок). Функцию syslog() можно использовать в обработчиках прерывания. Анализ текстов функций, осуществляющих форматированный ввод/вывод, показал, что они занимают достаточно много места.

Функция kprint() нестандартна. Для форматирования она использует функцию kvprintf(), которая является упрощенной версией функции vsprintf(). Собственно вывод производится функцией kwrite(). Существующая реализация функции kwrite(), содержащаяся в ППМ, выводит сообщения (только) на консоль побайтно в режиме опроса. Функция kprint() может быть использована в обработчиках прерываний и обладает наибольшей «живучестью» (в частности, не использует прерывания). Отметим, что использование режима опроса и запрещение прерываний функцией kwrite() ухудшает временные характеристики.

Для вывода сообщений ОС использует все три функции (kprint(), printf() и syslog()). В настоящее время использование функции printf() операционной системой приводит к тому, что в образ попадают функции форматирования и функции обслуживания потоков ввода и вывода (FILE*). То же самое относится и к функции syslog(). Для экономии памяти предлагается следующий подход. Нужно дать возможность пользователю при конфигурировании указать, что следует использовать сокращенный вариант функций форматирования. В этом случае функции printf() и syslog() будут использовать для форматирования функцию kvprintf().

При отладке сообщения обычно выводятся на терминал; при реальной эксплуатации терминала может не быть и для вывода сообщений лучше использовать флэш-память. Это обусловлено тем, что при работе без терминала сообщения, выводимые на терминал функциями kprint() и printf(), просто теряются. Функция syslog() поддерживает вывод сообщений во флэш память.

В силу описанного выше, предлагается для вывода сообщений в ОС явно использовать только функцию syslog(), доработав ее таким образом, чтобы при конфигурировании можно было указать форматирование в упрощенном формате для функции syslog(). Сообщения должны выводиться, минуя очередь, в следующих случаях:

– если при конфигурировании было указано, что сообщения данного уровня строгости следует выводить, минуя очередь (по умолчанию LOG_EMERG, LOG_ALERT и LOG_CRIT);

– если установлен флаг LOG_NODAEMON в аргументе priority при вызове функции syslog();

– при обработке ошибок, возникших при обработке ошибок.

При выводе на консоль, минуя очередь, следует использовать функцию kwrite() (вместо write()), эту возможность следует сделать доступной при конфигурировании ОСРВ.

Также при конфигурировании должна быть обеспечена возможность указать:

– какого уровня строгости сообщения не следует выводить (они будут просто теряться);

– какого уровня строгости сообщения следует выводить, минуя очередь.

Замена функций форматированного ввода/вывода на их облегченный аналог позволит

получить экономию памяти при уменьшении возможностей форматирования.

Средства отладки. В ОСРВ Багет 2.x к средствам отладки относятся:

– командный интерпретатор,

– локальный отладчик,

– удаленный отладчик,

– динамический загрузчик.

Кроме того, к средствам отладки также относятся информационные функции (команды), используемые при отладке и исполняемые с помощью командного интерпретатора или из программы. Эти команды можно разбить на следующие группы:

– команды получения информации об объектах ОС;

– команды вывода и изменения содержимого памяти;

– команды вывода информации о сети;

– средства проверки объектов ОС;

– команды управления выполнением программы;

– команды управления переменными окружения.

Отметим, что управление локальным отладчиком и динамическим загрузчиком осуществляется либо с помощью командного интерпретатора (команды вводятся в командный интерпретатор), либо из агента удаленной отладки.

В ОСРВ Багет 2.x на целевую ЭВМ загружаются только секции с программным кодом и проинициализированными данными. Для экономии оперативной памяти секция с таблицей имен не загружается. Чтобы иметь возможность использовать имена функций и переменных на целевой ЭВМ, при создании целевого образа формируется таблица символов, которая содержит сведения о внешних переменных и функциях операционной системы и прикладных программ, включенных в конкретный целевой образ. Один элемент таблицы описывает одну переменную или функцию, и позволяет работать с ними, указав имя функции или переменной. Таблица символов в основном используется командным интерпретатором. Для экономии оперативной памяти предлагается полную таблицу символов заменить на сокращенную, которая включает в себя только команды, описанные в документации, или только список команд и функций, заданных при конфигурировании целевого образа. Можно также полностью исключить таблицу символов из образа. В последнем случае исключаются и функции работы с таблицей имен.

Командный интерпретатор предназначен для интерактивной работы с ОС и прикладной программой. Для взаимодействия с интерпретатором – ввода команд и получения результатов – используется консоль. В качестве консоли может быть использован алфавитно-цифровой терминал или telnet-соединение. Командный интерпретатор является реентерабельной программой и поэтому можно использовать один экземпляр интерпретатора для работы с ним из различных консолей. Если из конфигурации исключается таблица имен, тогда командный интерпретатор становится ненужным и его также можно исключить.

Локальный отладчик включает в себя установку/удаление отладочных точек останова, пошаговое выполнение отлаживаемой программы, выполнение заявок агента удаленной отладки. Исключение локального отладчика не влечет исключение дизассемблера и средств раскрутки стека.

Если дизассемблер не будет включен, то локальный отладчик будет работать без использования дизассемблера (будет печатать шестнадцатеричный код команды вместо ее псевдо ассемблерного представления).

При исключении раскрутки стека, эта функция будет заменена заглушкой.

Использование динамического загрузчика позволяет не включать отлаживаемый модуль в образ системы. Достаточно его откомпилировать и затем загрузить с помощью динамического загрузчика. Динамический загрузчик требует включения таблицы имен и файловой системы.

Включение средств удаленной отладки требует поддержки сетевых средств и протокола RPC и, в существующей версии, средств динамической загрузки. Предлагается сделать динамический загрузчик необязательным для удаленного отладчика. Это уберет зависимость удаленного отладчика от таблицы символов.

Получить информацию об объектах ОС можно с помощью следующих команд:

– bufstat – вывод информации об использовании буферов;

– cnd – вывод информации об условной переменной;

– io – вывод списка устройств;

– lkup – вывод содержимого таблицы имен;

– mq – вывод информации об очереди сообщений;

– mtx – вывод информации о мьютексе;

– sem – вывод информации о семафоре

– tmr – вывод информации о таймере;

– vnstat – вывод информации об использовании структур vnode.

Если команды вывода информации об объектах ОС включены образ ОСРВ, то каждая команда из этого списка попадает в целевой образ только при условии, что в образе содержатся объекты, к которым относится эта команда. Включение информационных команд требует наличие командного интерпретатора.

Команды вывода и изменения содержимого памяти могут быть исключены из целевого образа.

Команды вывода информации о сети требуют наличие сетевых средств.

Включение в целевой образ средств проверки объектов ОС позволяют из командного интерпретатора или из программы вызывать функцию проверки на корректность заданных или всех объектов ОС.

К командам управления выполнением программы относятся команды управления потоками и команда повторной загрузки операционной системы:

– ti - вывод сведений о состоянии потоков управления;

– tst – создание потока управления;

– tc – удаление потока управления;

– ts – приостановка потока управления;

– tr – продолжение работы потока управления;

– tk – посылка сигнала потоку управления;

– r – повторная загрузка системы.

К командам управления переменным окружения относятся следующие команды:

– setenv – установка значения переменной окружения;

– unsetenv – отмена определения переменной окружения;

– printenv – вывод значений переменных окружения.

Любую группу команд отладки можно включать и/или исключать из целевого образа.

Условные переменные. ОСРВ не использует явно условные переменные, но ППМ может их использовать (например, в драйвере устройства «манчестер»).

Исключение средств поддержки условных переменных в ОС дает возможность не создавать класс условных переменных. Функции поддержки условных переменных заменяются заглушками.

Очереди сообщений предназначены для обмена данными между потоками управления. Очереди сообщений обеспечивают две основные операции:

– отправка сообщения в очередь (функции mq_send() и mq_timedsend),

– прием сообщения из очереди (функции mq_receive() и mq_timedreceive).

Очереди сообщений используются потоком cron, который в свою очередь использует средствами поддержки сети. В силу этого, если поддержка очередей сообщений не включена в ОС, то поддержка сети не может быть включена в ОС.

Очереди сообщений используются также потоком локального отладчика. В силу этого, если поддержка очередей сообщений не включена в ОС, то поддержка локального отладчика не может быть включена в ОС.

Программные таймеры позволяют запланировать выполнение какой-либо деятельности в определенный момент времени в будущем. Для создания таймера используется функция timer_create(). Одним из аргументов этой функции является структура sigevent, которая определяет вид оповещения о срабатывании таймера (например, посылка сигнала или выполнение указанной функции).

Если указан период срабатывания таймера, то после каждого срабатывания таймера он будет заново установлен и запущен со значением, равным указанному периоду.

Ни ОС, ни ППМ не используют явно таймеры.

Невключение средств поддержки таймеров в ОС дает возможность не создавать класс таймеров. Функции поддержки таймеров заменяются заглушками, которые дают код ошибки ENOSYS.

Для обеспечения периодической обработки прикладная программа может использовать функцию nanosleep() вместо таймеров.

Региональные настройки (локализация). При отключении средств поддержки региональных настроек вместо них в образ ОС включаются сокращенные варианты функций setlocale() и localeconv(). Функция setlocale() возвращает код ENOSYS, а функция localeconv() возвращает NULL. Все функции, зависящие от локализации, будут работать также, как и с локализацией C@short.

Кроме того, можно отключить средства печати сообщений об ошибках: функции strerror() и perror(). Эти функции заменяются заглушками, которые не выполняют никаких действий. Функция порождает код ошибки ENOSYS.

Программные каналы. Ни ОСРВ, ни ППМ не используют явно программные каналы.

Средства их поддержки не конфигурируются, т. к. они попадают в целевой образ только при использовании их прикладной программой.

Сигналы. Для исключения функций поддержки сигналов используется модуль с пустыми функциями («заглушками»). Эти функции возвращают код ошибки ENOSYS. Если прикладная программа непосредственно вызывает функции поддержки сигналов, то по коду ошибки она может определить, что функции поддержки сигналов не включены в ОС.

ОС может посылать сигналы по запросу прикладной программы:

– для оповещения о срабатывании таймера;

– для оповещения о завершении асинхронной операции ввода/вывода (aio);

– для оповещения о поступлении сообщения в пустую очередь сообщений;

– при вводе некоторых символов с терминала;

– при обработке исключительных ситуаций с плавающей точкой по стандарту IEEE754;

– при возникновении исключительной ситуации (если это запрошено при конфигурировании).

Если при создании таймера функцией timer_create() запрашивается уведомление с помощью сигналов и сигналы не были включены при конфигурировании, то таймер не создается с кодом ошибки ENOSYS.

Если при вызове функций aio_write(), aio_read() и lio_listio() запрашивается уведомление с помощью сигналов и сигналы не были включены при конфигурировании, то операция не выполняется (код ошибки ENOSYS).

Если при вызове функции mq_notify() запрашивается уведомление с помощью сигналов и сигналы не были включены при конфигурировании, то операция не выполняется (код ошибки ENOSYS).

Если средства поддержки сигналов не были включены при конфигурировании, то при вводе любых символов с терминала ОС не порождает сигналов.

Если средства поддержки сигналов не были включены при конфигурировании, то средства конфигурирования не позволяют подключить средства обработки исключительных ситуаций команд плавающей арифметики по стандарту IEEE754. В этом случае ошибки при выполнении операций с плавающей точкой обрабатываются так же, как и другие ошибки.

Если средства поддержки сигналов не были включены при конфигурировании, то средства конфигурирования не позволяют обрабатывать исключительные ситуации путем посылки сигналов.

Семафоры. ОС не использует явно семафоры, но ППМ их активно использует (это основной способ уведомления потока управления из обработчика прерываний).

Исключение семафоров даёт возможность не создавать класс семафоров. Функции поддержки семафоров заменяются заглушками, которые дают код ошибки ENOSYS,

Заключение

В работе проанализированы отдельные части ОСРВ.

Выделены те, которые обязательно должны присутствовать в целевом образе, те, которые могут быть исключены, если они не используются прикладной программой или ППМ, и те, которые могут быть существенно уменьшены.

Разработана методика создания компактной ОСРВ на основе масштабирования уже существующей ОС.

Описанный подход позволяет получить компактную ОСРВ высокой надежности и со сравнительно небольшими затратами на разработку.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) "Исследование принципов построения компактной операционной системы для отечественных радиационно-стойких процессоров." (0065-2016-0003)

Methods for reducing the resource consumption of the hard real-time operating system.

A.A. Asonov, A.N. Godunov, V.A. Soldatov

Abstract: Despite the general growth in the amount of available RAM, there is a number of tasks that operate under hard real-time conditions, for which RAM is a limited resource. In such tasks, restrictions are imposed on resources used by a real-time operating system (RTOS). To reduce the memory size required for the RTOS, it is suggested to execute the code of the target system straight from ROM and enhance scalability of the existing OS. This approach allows to reduce the software development efforts hereby providing high reliability of RTOS. Some components of the RTOS (interrupts and exceptions handling, multithreading, time service, memory allocation) are essential and cannot be completely removed from the target image, but can only be reduced, to some extent. All other components of the operating system can be either significantly reduced, or completely removed, if they are not used by the application program or the board support package (BSP). These include networking, file systems, block devices, basic I/O operations, debugging tools, conditional variables, message queues, timers, regional settings, program channels, signals, semaphores.

Keywords: RTOS, interrupt handling, exception handling, scaling, drivers, configuration, RTOS Baget

Литература

1. С.Г.Бобков, Импортозамещение элементной базы вычислительных систем – “Вестник Российской Академии Наук”, 2014, том 84, № 11, с. 1010–1016.

2. М.С.Горбунов. Современные микропроцессоры космического и специального назначения разработки ФГУ ФНЦ НИИСИ РАН, а также перспективные ЭВМ на их основе. Доклад на III-й межотраслевой конференции ОАО «НИИМЭ и Микрон», 2016 г.

3. Е.А.Новожилов. Микропроцессоры НИИСИ РАН для космического применения. Доклад на 2-ой Международной научной конференции «Интегральные схемы и электронные модули», Алушта, 26-30 сентября 2016 г.

4. Е.А.Новожилов. Реализация интерфейса SpaceWire в микропроцессорах НИИСИ РАН для космического применения. Доклад на 3-ей Международной научной конференции «ЭКБ и электронные модули», Алушта, 2-7 октября 2017 г.

5. В.Л.Безруков, А.Н.Годунов, П.Е.Назаров, В.А.Солдатов, И.Г.Хоменков. Введение в ос2000 // Вопросы кибернетики, М.: НИИСИ РАН, 1999 г., с. 76-106.

6. А.Н.Годунов, В.А.Солдатов. Операционные системы семейства Багет (сходства, отличия и перспективы) – «Программирование», Москва, 2014, №5, с. 68-76.

7. А.Н.Годунов, В.А.Солдатов, Конфигурирование многопроцессорных систем в операционной системе реального времени Багет – «Программная инженерия», Москва, 2016, №6, с. 243-251.

8. Е.А.Герлиц, В.В.Кулямин, А.В.Максимов, А.К.Петренко, А.В.Хорошилов, А.В.Цыварев. Тестирование операционных систем // Труды Института системного программирования РАН. Том 26, выпуск 1, 2014 г.

Поддержка версий изделий при автоматизации производственного процесса

А.Б. Бетелин

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: ab@niisi.msk.ru

Аннотация: В статье описывается простой программный механизм поддержки версий изделий, применяющийся при автоматизации деятельности производственного подразделения.

Ключевые слова: версии изделий, производство, автоматизация, программное обеспечение, база данных.

1. Введение

В статье [1] было приведено общее описание программного обеспечения, предназначенного для автоматизации решения учетных и управленческих задач в подразделении, производящем экспериментальные образцы электронно-вычислительной аппаратуры, в том числе кабельную продукцию.

Каждое из этих изделий изготавливается на основании конструкторской документации (КД) на него, которая хранится в архиве КД.

Случается так, что изделие по тем или иным причинам модернизируется.

Например, разработчик изделия может принять решение вместо одной марки провода применить другую, изменить длину кабеля, добавить в изделие дополнительные разъемы и т.д.

В результате подобных решений вносятся изменения в КД на изделие и, в том числе, в его спецификацию.

Таким образом, по мере модернизации изделия последовательно появляются различные его версии, что отражается в появлении версий КД на изделие и, в частности, версий спецификации этого изделия.

Поскольку в подсистеме АСУ НИИСИ РАН для автоматизации деятельности производственного подразделения [1] КД на изделие представлено единственным объектом типа "спецификация изделия", то в дальнейшем нас будут интересовать именно версии спецификаций и их отображение на таблицы реляционной базы данных (БД).

2. Спецификации изделий и их версии

Потребность в комплектующих, необходимых для изготовления изделия, определяется на основании его спецификации. В подсистеме автоматизации производственной деятельности [1] спецификация изделия хранится в двух таблицах БД - таблице спецификаций и таблице комплектации. Запись (строка) таблицы спецификаций, представляющая объект типа "спецификация", содержит наименование изделия, его обозначение и некоторую дополнительную информацию:

```
CREATE TABLE t_s
(
   id INT UNSIGNED PRIMARY KEY
AUTO_INCREMENT COMMENT 'Уникальный
идентификатор записи',
   naim_i VARCHAR(128) COMMENT
'Наименование изделия',
   obozn_i VARCHAR(40) COMMENT
'Обозначение изделия',
   ...
   UNIQUE KEY (obozn_i)
) COMMENT 'Таблица спецификаций';
```

Здесь и далее в описаниях таблиц многоточием заменены информационные поля, не имеющие значения для рассматриваемой темы. Со строкой таблицы спецификаций связаны несколько записей (строк) таблицы комплектации, образующих перечень входящих в изделие комплектующих. Привязка комплектующего к спецификации осуществляется при помощи поля spec строки таблицы комплектации, куда заносится ссылка на соответствующую строку таблицы спецификаций:

```
    CREATE TABLE t_k
(
   id INT UNSIGNED PRIMARY KEY
AUTO_INCREMENT COMMENT 'Уникальный
идентификатор записи',
   naim_k VARCHAR(255) COMMENT
'Наименование комплектующего',
   kol_k DECIMAL(18,2) COMMENT
'Количество комплектующих',
   spec INT UNSIGNED COMMENT 'Ссылка
на спецификацию',
   ...
   FOREIGN KEY (spec) REFERENCES t_s
(id)
) COMMENT 'Таблица комплектации';
```

Рассмотрим далее, в чем может состоять отличие двух версий одной и той же спецификации.

В соответствии со стандартами [2,3] наименование и обозначение присваиваются изделию в начале его разработки и в дальнейшем меняться не должны. По этой причине две версии спецификации одного и того же изделия не могут иметь разные наименования или обозначения. Стало быть, различия между ними могут обнаружиться лишь в наборах записей о комплектующих - в количестве записей и значениях некоторых полей в них.

Если в состав изделия добавляется новое комплектующее, то в соответствующий набор записей о комплектующих добавляется новая запись. Если из состава изделия удаляется какое-то комплектующее, то из соответствующего набора записей о комплектующих удаляется ранее существовавшая запись. В случае замены одного комплектующего на другое или же изменения их количества происходит модификация ранее существовавшей записи - в ней изменяются значения полей "наименование" или "количество".

Таким образом, все различия между версиями одной спецификации сосредоточены в пределах таблицы комплектации.

3. Хранение версий спецификаций в базе данных

Чтобы предоставить возможность доступа к различным версиям одной спецификации, необходимо обеспечить фиксацию в БД различных вариантов соответствующего набора записей о комплектующих. Для этого, во-первых, должны сохраняться все записи о комплектующих, которые были бы удалены в при переходе от предыдущей версии спецификации к последующей. Во-вторых, при изменении существующей записи о комплектующем необходимо сохранить ее текущее состояние, создать ее копию и уже в эту копию внести новые значения изменяемых полей.

В качестве простейшего способа хранения разных версий одной спецификации в БД можно было бы предложить, например, следующий. Для каждой версии спецификации создается отдельная запись в таблице спецификаций. Аналогично для каждого комплектующего, относящегося к некоторой версии спецификации, создается отдельная запись в таблице комплектации. Нетрудно видеть, что при таком подходе образуется большое число избыточных записей в таблицах БД. Например, если спецификация содержит несколько десятков комплектующих, а две ее версии отличаются только одним комплектующим, то абсолютное большинство записей о комплектующих для новой версии будут дубликатами записей для предыдущей версии, уже имеющихся в таблице комплектации.

Однако хотелось бы решить задачу хранения версий, исключив появление каких-либо избыточных записей в обеих таблицах. Общие случаи хранения версий объектов в БД и отслеживания истории их изменений уже рассматривались в литературе [4-9]. Этот вопрос относится к области *медленно меняющихся измерений* (Slowly Changing Dimensions, SCD) [4,5], изучающей методы мониторинга изменений данных в БД. Такие задачи решаются, как правило, с помощью понятия медленно меняющихся измерений второго типа (SCD Type 2). В этом случае к записи БД, представляющей изменяющийся объект, добавляются дополнительные поля, в которых фиксируются моменты времени, когда запись стала актуальной и когда она перестала быть таковой. Также к записи могут быть добавлены индикаторы, показывающие, активна ли запись, была ли она удалена и т.п. При каждом изменении записи она копируется, предыдущая ее версия сохраняется в БД и заполняются все дополнительные поля для текущей и предыдущей версий.

В производственном процессе работа со спецификациями изделий имеет свои особенности, позволяющие упростить хранение их версий по сравнению с общим случаем.

При поступлении заявки на изготовление изделия используется текущая, т.е. самая последняя, версия его спецификации. Момент времени ее появления для

производства не важен, существенно лишь то, что эта версия является самой "свежей". Поэтому вместо фиксации этих моментов времени можно запоминать только номер версии спецификации.

Кроме того, для поддержки производственного процесса не требуется сохранять абсолютно все когда-либо существовавшие версии спецификации.

Допустим, что было изготовлено какое-то количество изделий в соответствии с некоторой спецификацией. Затем произошла модернизация изделия, в результате чего появилась новая версия его спецификации и соответственно изменился список его комплектующих. Поскольку предыдущая версия спецификации использовалась при производстве изделий, запись о ней наряду с соответствующими записями о комплектующих необходимо сохранить в БД для обеспечения корректности данных.

Далее, если по новой версии спецификации до следующей модернизации не было изготовлено ни одного изделия, то информация об этой версии спецификации не представляет интереса для производства. Поэтому при следующей модернизации записи о комплектующих могут быть просто обновлены без сохранения их предыдущего состояния в БД.

В подсистеме автоматизации производственной деятельности [1] заявки на изготовление изделий хранятся в таблице заявок, каждая запись которой содержит, помимо прочей информации, ссылку на спецификацию:

```
CREATE TABLE t_z
(
   id INT UNSIGNED PRIMARY KEY
AUTO_INCREMENT COMMENT 'Уникальный
идентификатор записи',
   nomer_z VARCHAR(20) COMMENT
'Номер заявки',
   spec INT UNSIGNED COMMENT 'Ссылка
на спецификацию',
   ...
   UNIQUE KEY (nomer_z),
   FOREIGN KEY (spec) REFERENCES t_s
(id)
) COMMENT 'Таблица заявок';
```

Таким образом, в БД необходимо сохранять только те версии спецификаций, на которые имеются ссылки из таблицы заявок.

Рассмотрим, каким образом можно было бы модифицировать приведенные выше таблицы БД, чтобы реализовать хранение версий спецификаций с учетом особенностей производственного процесса. Введем дополнительные поля для хранения номера версии в таблицы заявок, спецификаций и комплектации.

Определим номер версии как целое положительное число.

Когда в БД создается новая спецификация, будем считать, что ее версия имеет номер 1.

При модернизации изделия, т.е. при появлении следующей версии спецификации, поступим следующим образом.

Если по этой спецификации было изготовлено хотя бы одно изделие, то сохраним в БД предыдущую версию, а номер текущей увеличим на 1.

Если же ни одного изделия изготовлено не было, то произведем необходимые изменения в записях о комплектующих, не сохраняя их предыдущего состояния и не увеличивая номер версии.

В результате в таблице комплектации для каждой спецификации будут присутствовать только строки ее текущей версии, или же строки ее предыдущих версий, по которым когда-либо изготавливались изделия.

Описания модифицированных таблиц, с учетом сказанного выше, имеют следующий вид:

```
CREATE TABLE t_z
(
   id INT UNSIGNED PRIMARY KEY
AUTO_INCREMENT COMMENT 'Уникальный
идентификатор записи',
   nomer_z VARCHAR(20) COMMENT
'Номер заявки',
   spec INT UNSIGNED COMMENT 'Ссылка
на спецификацию',
   nv INT UNSIGNED COMMENT 'Номер
примененной версии спецификации',
   ...
   UNIQUE KEY (nomer_z),
   FOREIGN KEY (spec) REFERENCES t_s
(id)
) COMMENT 'Таблица заявок';


CREATE TABLE t_s
(
   id INT UNSIGNED PRIMARY KEY
AUTO_INCREMENT COMMENT 'Уникальный
идентификатор записи',
   naim_i VARCHAR(128) COMMENT
'Наименование изделия',
   obozn_i VARCHAR(40) COMMENT
'Обозначение изделия',
   nv_cur INT UNSIGNED COMMENT
'Номер текущей версии спецификации',
   ...
   UNIQUE KEY (obozn_i)
) COMMENT 'Таблица спецификаций';
```

```
CREATE TABLE t_k
(
   id INT UNSIGNED PRIMARY KEY
AUTO_INCREMENT COMMENT 'Уникальный
идентификатор записи',
   naim_k VARCHAR(255) COMMENT
'Наименование комплектующего',
   kol_k DECIMAL(18,2) COMMENT
'Количество комплектующих',
   spec INT UNSIGNED COMMENT 'Ссылка
на спецификацию',
   nv_begin INT UNSIGNED COMMENT
'Номер начальной версии специфика-
ции',
   nv_end INT UNSIGNED COMMENT
'Номер конечной версии специфика-
ции',
   ...
   FOREIGN KEY (spec) REFERENCES t_s
(id)
) COMMENT 'Таблица комплектации';
```

Опишем подробнее, что происходит с записями в таблицах и каким образом модифицируются значения полей при создании и изменении спецификации изделия.

Итак, когда создается новая спецификация, полю *nv_cur* в строке таблицы спецификаций, а также полям *nv_begin* и *nv_end* в соответствующих строках таблицы комплектации присваивается значение 1.

Когда происходит модернизация изделия и появляется следующая версия спецификации, возможны две ситуации.

Если по спецификации, требующей изменения, было изготовлено хотя бы одно изделие, то значение поля *nv_cur* строки таблицы спецификаций следует увеличить на 1. Далее необходимо обновить соответствующий набор записей о комплектующих, сохранив его предыдущую версию:

– если запись из этого набора без изменений переходит в новую версию, то полю *nv_end* присваивается значение поля *nv_cur* из строки таблицы спецификаций;

– если к набору добавляется новая запись, то ее полям *nv_begin* и *nv_end* присваивается значение поля *nv_cur* из строки таблицы спецификаций;

– если изменяется существующая запись, то она копируется, затем требующим модификации полям этой копии присваиваются новые значения, а поля *nv_begin* и *nv_end* получают значение поля *nv_cur* из строки таблицы спецификаций;

– если запись подлежит удалению из набора, то никаких действий с ней не производится.

Если же по этой спецификации ни одного изделия изготовлено не было, то значение поля *nv_cur* строки таблицы спецификаций остается прежним, а соответствующий набор записей о комплектующих обновляется следующим образом:

– если запись из этого набора без изменений переходит в новую версию, то никаких действий с ней не производится;

– если к набору добавляется новая запись, то ее полям *nv_begin* и *nv_end* присваивается значение поля *nv_cur* из строки таблицы спецификаций;

– если изменяется существующая запись, то требующим модификации полям присваиваются новые значения;

– если запись подлежит удалению из набора, то при равенстве значений полей *nv_begin* и *nv_end* она удаляется, в противном же случае никаких действий с ней не производится.

Теперь рассмотрим, каким образом становится возможным извлекать информацию, относящуюся к определенной версии некоторой спецификации.

Пусть необходимо сформировать заявку на изготовление изделия и определить, какие комплектующие необходимы для его изготовления. Спецификацию изделия будем идентифицировать, например, его обозначением.

Т.к. номер текущей (последней) версии спецификации содержится в поле *nv_cur* строки таблицы спецификаций, то соответствующие этому номеру строки таблицы комплектации будут возвращены в качестве результата такого запроса:

```
SELECT * FROM t_k WHERE
  spec =
    (SELECT id FROM t_s WHERE
     obozn_i = 'Обозначение_изделия'
    )
  AND
  nv_end =
    (SELECT nv_cur FROM t_s WHERE
     obozn_i = 'Обозначение_изделия'
    );
```

Допустим теперь, что по одной из предыдущих версий спецификации было изготовлено некоторое количество изделий.

Как определить актуальный на тот момент времени набор комплектующих? Воспользуемся тем, что номер версии спецификации автоматически фиксируется в поле *nv* соответствующей строки таблицы заявок при формировании заявки за изготовление.

Заявку будем идентифицировать, например, ее номером. Тогда строки таблицы комплектации, соответствующие актуальной на тот момент версии

спецификации, можно извлечь при помощи следующего запроса:

```
SELECT * FROM t_k WHERE
  spec =
    (SELECT spec FROM t_z WHERE
     nomer_z = 'Номер_заявки'
    )
  AND
    ( (SELECT nv FROM t_z WHERE
       nomer_z = 'Номер_заявки'
      )
      BETWEEN
      nv_begin AND nv_end
    );
```

4. Заключение

В статье изложен подход к программной поддержке версий изделий, обеспечивающий минимальные накладные расходы на хранение данных.

Описанный здесь простой механизм реализован и успешно функционирует в подсистеме АСУ НИИСИ РАН для автоматизации деятельности производственного подразделения.

.

Support for versions of products for automation of the production process

A.B. Betelin

Abstract. This article describes a simple software mechanism to support versions of products for automation of the manufacturing division.

Keywords: versions of the products, manufacturing, automation, software, database.

Литература

1. А.Б.Бетелин. Программное обеспечение для автоматизации учетно-управленческой деятельности подразделения по изготовлению единичной продукции. "Труды НИИСИ РАН", т.5 (2015), №2, 79-85.

2. ГОСТ 2.104-2006 "Единая система конструкторской документации. Основные надписи". Москва, Стандартинформ, 2007. http://www.internet-law.ru/gosts/gost/974/ (30.08.2017)

3. ГОСТ 2.201-80 "Единая система конструкторской документации. Обозначение изделий и конструкторских документов". Москва, Государственный комитет СССР по стандартам, 1980. http://www.internet-law.ru/gosts/gost/23473/ (30.08.2017)

4. М.В.Ковтун. Медленно меняющиеся измерения (Slowly Changing Dimensions) в корпоративном хранилище данных. Октябрь 2010. https://www.prj-exp.ru/dwh/slowly_changing_dimension.php (30.08.2017)

5. С.Малакшинов. Версионность и история данных. 12.08.2010. https://habrahabr.ru/post/101544/ (30.08.2017)

6. А.Г.Прилипко. Хранение истории изменения информации в реляционной базе данных. "Труды НИИСИ РАН", т.2 (2012), №1, 67-72.

7. В.Федотов, С.Рихтер, Ф.Энге. SETVERS - версионирование наборов объектов. "Вестник МГСУ", №6 (2011), 374-383.

8. How to store different version of data in a relational database? 19.12.2012. https://stackoverflow.com/questions/9477359/how-to-store-different-version-of-data-in-a-relational-database (30.08.2017)

9. Versioned Data. 29.11.2009. http://blog.schauderhaft.de/2009/11/29/versioned-data/ (30.08.2017)

Развитие генератора случайных тестов tergen

А.С. Куцаев

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: koutsaev@niisi.msk.ru

Аннотация: Качество случайных тестов зависит от связи между тестовыми ситуациями и возможностями случайного выбора. Использование моделей устройств процессора делает эту зависимость более явной. Также содержательные случайные тесты можно получать на основе детерминированных, используя случайный выбор и такие инструменты генератора тестов, как переменные и встроенные функции. В частности, эти инструменты позволяют реализовать интенсивное тестирование с вызовами процедур. Новые возможности для тестирования открывает также встроенный симулятор. Он облегчает контроль корректности, позволяет манипулировать содержимым регистров и памяти привычным для программиста способом и дает возможность создания случайных тестов с самопроверкой.

Ключевые слова: случайные тесты, шаблон, симулятор, переменные, процедуры.

1. Введение

Генераторы случайных тестов давно и успешно используются при тестировании микропроцессоров [1, 2, 3]. Для получения тестов большого объема в них достаточно задать простой шаблон и вероятности выбора инструкций и их аргументов, отвечающие цели тестирования. Однако частота тестовых ситуаций в таких тестах ограничена, поскольку каждый набор вероятностей обычно ориентирован на создание одной-двух "угловых" ситуаций, которые возникают тем реже, чем больше совпадающих условий для этого требуется.

Более содержательные тесты можно получить, используя в генераторах тестов модели устройств, таких как кэш или конвейер [4]. Модели позволяют управлять созданием специфических тестовых ситуаций. Например, модель кэша данных следит за состоянием каждой строки кэша. При выборе адреса для очередного обращения к памяти генератор тестов случайно выбирает ситуацию (промах с жертвой, повторное попадание и т.д.) и подбирает строку кэша и адрес, который приведет к выбранной ситуации.

Содержательность тестов связана с заданием адресов через регистры. Чтобы обеспечить корректное содержимое регистров, генератор добавляет в тест инструкции, не связанные с целью тестирования. Встроенный симулятор [4] помогает снизить их число, поскольку он контролирует содержимое регистров.

Тем не менее, при случайном выборе инструкций и аргументов частота создания тестовых ситуаций невелика. Из-за этого иногда предпочитают создавать случайные тесты на основе детерминированных, с включением в них случайного выбора. Для создания таких тестов требуются инструменты, такие как переменные и встроенные функции. Переменные служат для выбора инструкций и задания их аргументов, а функции обеспечивают действия над переменными, в том числе случайный выбор значений.

С помощью переменных случайные значения можно использовать в инструкциях и операторах наряду с константами. При этом программист может управлять содержимым регистров с помощью привычных инструкций.

Переменные и функции расширяют возможности, связанные с вызовом процедур. Вызов процедур используется во всех содержательных приложениях, а с помощью стохастического тестирования можно воспроизводить соответствующие типичные ситуации.

В отличие от традиционных симуляторов [5], в генераторе тестов к симулятору предъявляются особые требования, такие как выполнение кода по мере его генерации, замена ошибочных инструкций и отложенное выполнение.

В то же время для генератора достаточно ограниченного моделирования выполнения, при котором допускается неизвестное содержимое регистров и памяти [4].

В частности, симулятор поддерживает переходы с неизвестным условием в т.н. режиме условного выполнения.

Цель настоящей работы - описание новых возможностей, которые дает генератор случайных тестов tergen, разработанный в НИИСИ РАН для процессоров архитектуры MIPS.

В Главе 2 дано описание переменных языка программы построения теста. Глава 3 посвящена встроенным функциям. В Главе 4 представлен подход к тестированию вызовов процедур, использующий переменные и функции. В Главе 5 описываются свойства встроенного симулятора. В Заключении приводятся выводы и рассматриваются возможности развития генератора тестов tergen.

2. Переменные

Аналоги переменных встречаются даже в простых генераторах случайных тестов. Так, при использовании итератора с последовательной подстановкой элементов списка ссылка на текущий элемент списка по сути является переменной.

Такая переменная может использоваться при выборочном задании аргументов инструкций.

Использование полноценных переменных в случайных тестах позволяет повысить гибкость программирования, так как случайный выбор значений отделен от генерации инструкций.

Так, если нужно выполнить несколько обращений к памяти по разным случайным адресам, можно отдельно выбрать случайные значения баз адреса и регистры для них. Затем также отдельно выполняется загрузка баз в регистры и сами обращения к памяти.

Кроме действий над элементами массивов, предоставлены действия над массивами в целом. Содержимое массива можно задать или выбрать случайно по заданным свойствам, перемешать, устроить перебор элементов и т.д.

В генераторе случайных тестов tergen используются переменные базовых типов: литерная строка, число, регистр, инструкция и дерево инструкций с весами.

Дерево инструкций служит для случайного выбора инструкций, а переменная этого типа позволяет ссылаться на одно и то же дерево из разных мест программы.

Предоставлены также одномерные массивы базовых типов. Ссылка на массив чисел, регистров или инструкций может задавать список значений для соответствующего итератора.

Синтаксис описания переменных в шаблоне аналогичен языку Си.

Все переменные глобальны, т.е. действуют как в шаблоне, так и в макросах. Каждое описание начинает действовать сразу по прочтении.

Как следствие, описания в шаблоне действуют раньше описаний в макросах. Допускаются неявные описания меток и переменных в итераторах.

Повторное описание (исключая неявное) запрещено. Переменной и массиву можно присвоить значения как при их описании, так и позже.

Длина массива задается константой либо определяется в результате присваивания или вызова функции. Действия над массивом могут приводить к изменению его длины.

Оператор присваивания позволяет присвоить переменным значения в любом месте программы.

Простой переменной можно присвоить константу, значение другой переменной, в том числе элемент массива, и значение, возвращаемое функцией.

Присвоение массива массиву приводит к копированию содержимого и длины.

3. Встроенные функции

Функции в генераторе случайных тестов позволяют выполнять необходимые действия, которые нельзя или слишком сложно выполнить с помощью других операторов. Функции не пишутся пользователем, они встроены в генератор тестов. Набор функций определяется потребностями тестирования.

Некоторые функции вырабатывают значение, которое должно быть присвоено переменной. Другие функции просто выполняют действия над своими параметрами. Соответственно имеется два варианта синтаксиса. Опускать присваивание в первом случае, как это разрешено в языке Си, здесь нельзя, так как у функций отсутствует сколько-нибудь полезный побочный эффект. Вызов функции может приводить к добавлению инструкций в код, но чаще действует только на переменные.

Функции, предоставленные в текущей версии генератора тестов, делятся на следующие группы.

- Выбор числа и загрузка в регистр: rand, gload;
- Действия с любыми массивами: shuffle, rndcopy, segment;
- Действия с адресами данных: daddr, doffs;
- Действия с массивами регистров GPR: gselect, disable, enable, wrsubr;

В первой группе функция rand выбирает случайное целое число в заданных границах, а функция gload добавляет в код инструкции, которые загружают заданное значение в регистр общего назначения (аналог псевдоинструкции li). Пример:

```
int nn1;
register rr1 = r4;
...
// Выбрать и загрузить случайное число
nn1 = rand (0x1000, 0x1FFF);
gload (rr1, nn1)
```

Функции shuffle, rndcopy и segment выполняют действия с массивами любого типа. Функция shuffle случайно перемешивает заданный массив. Функция rndcopy возвращает массив, случайно выбранный из нескольких заданных. Функция segment возвращает подмассив заданного массива. Пример:

```
int nn1;
register ar1[10], ar2[10], ar3[10];
register ad1[], ad2[];
...
// Выбрать, перемешать, вырезать часть
ad1 = rndcopy (ar1, ar2, ar3);
shuffle (ad1);
ad2 = segment (ad1, 2, 7);
```

В этом примере случайно выбирается один из трех массивов ar1, ar2, ar3 и копируется в ad1. Затем содержимое ad1 случайно перемешивается. Наконец, подмассив от ad1[2] до ad1[7]

копируется в ad2. Те же функции можно использовать для массивов чисел или инструкций.

Функции daddr и doffs служат для работы со случайными адресами памяти данных. Функция daddr возвращает случайный адрес, для которого заданы выравнивание, разрешение на чтение/запись и запас расстояния до границ области памяти. Если адрес используется в цикле как базовый, то этот запас позволяет инкрементировать базу, не выходя за границу области. Функция doffs возвращает случайное 16-битовое смещение для заданной базы адреса, которая определяет и область памяти. Примеры для daddr и doffs см. далее, при описании процедур с многократным вызовом.

Следующие функции действуют над массивами регистров общего назначения (GPR). Функция gselect заполняет массив случайно выбранными различными регистрами, которые на момент вызова можно использовать для чтения/записи данных. Число регистров задается текущей длиной массива. Регистры r0 и r31 не выбираются никогда, последний - чтобы избежать конфликтов со случайными инструкциями вызова процедур. Не выбираются также регистры, в которые почему-либо запрещена запись.

Функция disable запрещает запись в регистры, находящиеся в заданном массиве, а функция enable отменяет такой запрет. Например, если нужно с помощью gselect выбрать две группы регистров, вызовы disable и enable обеспечат отсутствие пересечений в группах. Другие примеры см. далее, при описании процедур с многократным вызовом.

Еще одна функция, wrsubr, используется только в специальном макросе call (см. далее), который заполняет тела процедур для многократного вызова. Она назначает регистры, в которые разрешена случайная запись в процедурах. Она же ограничивает использование этих регистров вне тел процедур, так как при случайном вызове процедуры их содержимое может быть изменено.

4. Вызов процедур

4.1. Требования и возможности

Вызов процедур используется во всех содержательных приложениях. В стохастических тестах моделирование вызова процедур состоит в воспроизведении состава инструкций и действий в типичных ситуациях.

Действия при вызове процедур, такие как передача параметров и сохранение данных в стеке, обусловлены соглашениями для программного интерфейса. Вызов обычно включает копирование регистров в память и переход с возвратом, а завершение вызова - восстановление содержимого регистров из памяти и переход по адресу возврата.

При стохастическом тестировании требуется примерное соответствие состава инструкций для этих действий, однако в точном воспроизведении действий нет необходимости.

Более того, последнее нежелательно, так как уменьшает вариативность теста.

Вызовы процедур часто бывают вложенными. В стандартном MIPS-процессоре нет аппаратной поддержки вложенных вызовов, а тестирование их программной поддержки не относится к задачам стохастического тестирования.

Генератор случайных тестов tergen предлагает два подхода к вызову процедур.

Одноразовые процедуры проще в реализации, однако целевые адреса для них выбираются по тем же правилам, что и для обычных переходов.

Процедуры с многократным вызовом более сложны в использовании, но позволяют адресовать любую строку кэша инструкций.

Ниже рассмотрены возможности и требования для обоих подходов.

4.2. Одноразовые процедуры

Одноразовые процедуры представляют собой вариант линейного выполнения, дополненный передачей управления и возвратом.

Генератор тестов поддерживает режим одноразовых процедур, в котором вызовом процедуры считается любой переход с возвратом, сохраняющий адрес возврата в регистр ra (r31).

Тело процедуры состоит из инструкций между переходом и инструкцией возврата jr ra. Инструкции вызова и возврата добавляются в тестовый код из шаблона.

Состав инструкций в вызывающей программе и в телах процедур также определяется только программой построения теста.

Корректность последующих вызовов здесь обеспечивать не требуется.

Тем не менее, для успешной генерации даже однократного вызова процедуры нужно выполнить еще несколько условий, перечисленных ниже.

При вызове процедуры нужно обеспечить правильный случайный выбор целевого адреса перехода.

Он не может выбираться в текущей области кода, так как это приведет к ошибкам: после возврата будет выполнена часть кода перед целевым адресом, затем тело процедуры, снова возврат и т.д.

Генератор исправляет такие ошибки, но, чтобы они вообще не возникали, в шаблоне должно быть описано две или более области кода, доступные для используемых инструкций перехода.

Вероятность выбора "близкого" перехода должна быть установлена в 0, а выравнивание целевого адреса - в 4, 8 или 16 байт (не больше, чтобы не расходовать напрасно память).

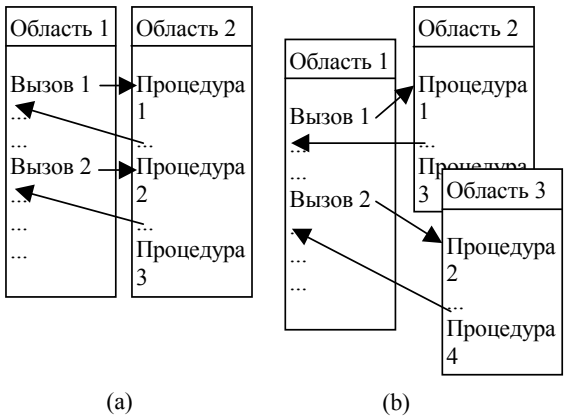

Рис1. Расположение вызовов и тел процедур в отсутствие посторонних переходов, (a) две области, (b) три области и более.

Если нет посторонних переходов, то при наличии двух областей кода в одной из них (базовой) будут только вызовы процедур, а в другой - только тела процедур и возвраты (Рис 1a). Если областей три и более, вызовы процедур также будут только в одной области (Рис. 1b). Если между вызовами процедур добавить обычные переходы со случайным выбором адреса, то вызовы и тела процедур распределятся по областям равномерно. В процедурах и вне их можно использовать циклы.

Адрес возврата хранится в регистре ra и не сохраняется автоматически в память. Чтобы он не был случайно затерт, в режиме одноразовых процедур генератор запрещает случайную запись в регистр ra после вызова процедуры и до выполнения возврата.

В приложениях вызовы процедур обычно составляют небольшую часть от обращений к памяти и вычислительных инструкций. При тестировании соотношение может меняться. Долю выполняемых переходов можно также увеличить путем добавления циклов.

4.3. Многократный вызов процедур

Многократный вызов процедуры предоставлен в генераторе тестов tergen в рамках использования модели кэша инструкций. Эта модель вырабатывает адреса перехода, которые вызывают в кэше такие тестовые ситуации как промахи и попадания с дополнительными свойствами. Чтобы для каждой строки кэша был достаточный выбор адресов процедур, в тестовом коде отводится подобласть, заполняемая телами процедур. Когда генератор тестов вырабатывает инструкцию перехода с возвратом, модель кэша выдает для нее адрес процедуры, отвечающий случайно выбранной тестовой ситуации в кэше. Другие переходы не используют эти адреса.

Пусть N_B - размер строки кэша в байтах, N_L - число строк в одном множестве кэша, а N_S - число множеств. Чтобы вызов процедуры всегда мог вызвать промах, нужно иметь (N_S+1) процедур на каждую строку. Общий размер подобласти будет тогда не менее N_B*N_L*(N_S+1), и для его минимизации размер процедуры нужно ограничить размером строки кэша.

Для генерации тел процедур должен быть предоставлен специальный макрос, имя которого задается в установке шаблона Macros:call.

Программа макроса должна содержать тела процедур, каждое из которых начинается с метки. Их количество может быть любым, так как вызов этого макроса происходит в особом режиме. В нем программа, начиная с первой метки, просматривается циклически, пока не будут сгенерированы все процедуры.

Число инструкций, которые генерируются после каждой метки, не должно быть меньше числа слов в строке кэша N_W = N_B/4. Когда очередные N_W инструкций записаны в память, генератор продолжает работу без записи, пока не встретит очередную метку или конец программы.

Инструкция возврата (jr ra) не генерируется автоматически. Она должна быть предусмотрена в макросе call в каждой процедуре. Возврат должен быть в предпоследнем слове процедуры или раньше.

Для использования процедур многократного вызова необходимо обеспечить корректность выполнения как процедур, так и основной программы между вызовами. Если то и другое вырабатывается случайно, нужно предусмотреть разделение ресурсов и действий между вызывающей программой и процедурами.

Проще всего обеспечить корректность, если в процедурах не меняется содержимое регистров и памяти.

Такие "пустышки" могут использоваться при отдельном тестировании кэша инструкций. Однако более интересно совместное тестирование кэша инструкций и, например, кэша данных. При этом типична ситуация, когда обращения к памяти данных происходят как как в вызывающей программе, так и в процедурах.

Основной проблемой при случайном обращении к памяти в процедурах является обеспечение нужного содержимого регистров базы адреса. Из-за небольшого объема одной процедуры загрузка базы в них оставляет мало места для других действий.

Для загрузки баз вне процедур генератор тестов tergen предоставляет следующие средства:

- Глобальные переменные типа число и регистр, позволяющие работать с регистрами базы и их содержимым в шаблоне и в макросе call.
- Функция gselect для случайного выбора регистров базы адреса.
- Функция daddr для случайного выбора значения базы.
- Функция disable для запрета случайной записи в регистры базы.

Перечисленные выше функции вызываются в шаблоне. При генерации тел процедур в макросе call используются также:

- Функция doffs для случайного выбора смещения по известному значению базы.
- Функция wrsubr для отведения регистров под случайную запись в процедурах.

Функция wrsubr разрешает случайную запись в процедурах только в регистры из указанного массива. Кроме того, wrsubr вводит запрет на использование этих регистров в вызывающей программе для хранения содержательных данных, таких как счетчик цикла, адрес данных и т.д., поскольку при любом вызове процедуры их содержимое может измениться непредсказуемо.

При обращении к памяти в процедурах формат задания инструкций зависит от того, где и как вырабатывается адрес и загружается база. Обычно регистр базы загружается в вызывающей программе, и тогда в процедуре инструкция обращения к памяти должна задаваться в формате ассемблера, т.е. без проверки адреса, который на момент вызова макроса еще не сгенерирован. Смещение (т.е. адрес) при этом может выбираться непосредственно перед обращением к памяти, при помощи функции doffs. При $NW = 8$ это позволит иметь до 7 обращений к памяти в одной процедуре. Другой формат инструкции, со случайным выбором и проверкой адреса, приведет к добавлению загрузки регистра базы, а на нее может уйти до 5 из 7 доступных инструкций. Нагрузка на кэш данных при этом окажется существенно ниже.

Приведем пример программ шаблона и макроса call при использовании многоразовых процедур. Описания приводятся в том порядке, в котором они действуют. Для обращений к памяти отводится четыре регистра базы, а для записи результатов выполнения инструкций в процедурах - еще два регистра. Остальные регистры свободно используются на чтение в процедурах и на чтение-запись - в среде вызова.

Программа шаблона начинается с описания общих переменных. Это значения базы адреса, которые вырабатываются в начале макроса call, и массив регистров, в которые будут записаны эти значения (см. далее первые две строки программы шаблона).

Чтение и вызов макроса call происходят в самом начале выполнения программы шаблона. Первыми читаются описания переменных, которые используются только в макросе. Затем, уже в ходе вызова, выполняется программа макроса:

```
:Program
register aw[2];          // Регистры на
запись
int ofv;              // Смещение адреса
// Случайный выбор регистров под базы
адреса
gselect (ab);
// Выработка четырех значений баз
bv0 = daddr (RW8,0,0);
bv1 = daddr (RW8,0,0);
bv2 = daddr (RW8,0,0);
bv3 = daddr (RW8,0,0);
disable (ab);        // Запрет записи
// Регистры на запись - выбор
gselect (aw);
// Назначение на запись в процедурах
wrsubr  (aw);
```

Вслед за начальными действиями происходит генерация тел процедур. Тело процедуры начинается с метки и заканчивается, когда число инструкций в процедуре отвечает размеру строки кэша (в примере - 8 слов). Далее генератор переходит к следующей метке либо к первой метке программы, пока не будут сгенерированы все процедуры.

Приведем пример одной процедуры с описанием действий. По причинам, отмеченным выше, для обращений к памяти данных используется формат ассемблера. Для других инструкций аргументы выбираются случайно. Каждое обращение к памяти использует свою базу и выработанные для этой базы адрес и смещение. Для тестирования может быть существенно, что инструкция, следующая за слотом задержки возврата, участвует в т.н. спекулятивном выполнении.

```
Entry0:                  // Метка начала
  ofv  =  doffs  (8,  bv0)  //  1:  выбор
адреса
  ld aw[0], ofv(ab[0]) // 1: загрузка
  ofv  =  doffs  (8,  bv1)  //  2:  выбор
адреса
  ld aw[1], ofv(ab[1]) // 2: загрузка
  ofv  =  doffs  (8,  bv2)  //  3:  выбор
адреса
  sd aw[0], ofv(ab[2]) // 3: сохранение
  ofv  =  doffs  (8,  bv3)  //  4:  выбор
адреса
  sd aw[1], ofv(ab[3]) // 4: сохранение
  sll()                              //  5:
произвольная
  jr r31                 // 6: возврат
  srl  ()                            //  7:
произвольная
  mthi()                             //  8:
спекулятивно
..mtlo()                 //    запасная
..mthi()                 //    запасная
Entry1:      // Метка следующего начала
```

Программа шаблона выполняется после вызова макроса call. Здесь функция gload загружает содержимое в регистры базы адреса, после чего случайная запись в них запрещается. В конце программы показано, как могут выглядеть вызовы процедур.

```
:Program
int   bv0,bv1,bv2,bv3;        //   Общие
переменные
register ab[4];

gload (ab[0], bv0);    // Загрузка
gload (ab[1], bv1);    // четырех баз
```

```
gload (ab[2], bv2);     //
gload (ab[3], bv3);     //
disable (ab);           // Запрет записи
repeat (10) {               // Вызовы -
итератор
  jal()                         // Вызов
процедуры
  add()                 // Произвольная
  sub()                 // Произвольная
}
:end
```

Заметим, что в вызывающей программе для обращений к памяти можно использовать любой формат инструкций. Здесь содержимое регистров базы известно генератору, а их перезагрузка запрещена вызовом функции disable.

При использовании процедур многократного вызова можно регулировать соотношение нагрузки на кэши. Нагрузка на кэш инструкций определяется частотой вызова процедур и в меньшей степени - подгрузкой инструкций основной последовательности. Нагрузка на кэш данных определяется частотой обращения к памяти. Арифметические инструкции можно использовать для выравнивания нагрузки.

5. Встроенный симулятор

Симулятор, встроенный в генератор случайных тестов, делает результаты выполнения инструкций известными на этапе их генерации, что открывает ряд новых возможностей. При генерации линейного участка тестового кода симулятор выполняет инструкции по одной. В случае перехода, связанного с циклом или вызовом процедуры, симулятор выполняет также инструкции, выработанные ранее, пока это возможно. Кроме того, симулятор должен обеспечивать отложенное выполнение инструкции, когда часть ее аргументов при генерации неизвестна. Такая необходимость возникает, например, когда в регистр нужно загрузить значение, которое вычисляется при использовании регистра, спустя несколько инструкций (оператор Preload).

Для генератора тестов в первую очередь должны быть известны исполнительные адреса кода и данных, которые вычисляются по содержимому регистров общего назначения (GPR). Оно зависит от выполнения вычислительных инструкций CPU, обменов с памятью данных, пересылок и переходов. Моделирование специальных действий, таких как обработка исключительных ситуаций или работа управляющего сопроцессора COP0, для генератора тестов большого интереса не представляет, но значительно увеличивает объем необходимых данных и требования к симулятору. Получение указанных данных может вызывать сложности, если составляющие системы разрабатываются независимо, а зависимость теста от этих данных будет ограничивать его применимость. Решением проблемы может быть ограниченное моделирование выполнения тестового кода, при котором допускается неизвестное содержимое части регистров и слов памяти.

Иногда способ ограниченного моделирования очевиден. Например, если содержимое регистра неизвестно, то результат операции, в которой этот регистр задает входной операнд, также неизвестен. Есть частные случаи, в которых результат не зависит от входного операнда - например, сравнение регистра с собой. Они могут учитываться, если результат важен (например, при вычислении условия перехода), но вообще доля операций с тождественным результатом очень невелика.

Если операнды известны, но при выполнении операции возникает исключительная ситуация, результат также может быть объявлен неизвестным. Как правило, это происходит при случайном выборе инструкций и аргументов, то есть когда результат не предназначен для использования с определенной целью.

Более сложный случай возникает, когда не удается вычислить условие перехода при его генерации. Генератор всегда старается обойтись без отмены выбранной инструкции. Здесь можно потребовать, чтобы значение условия перехода не влияло на состояние системы по достижении целевого адреса. Для этого генератор тестов выбирает переход на несколько инструкций вперед и заполняет отрезок до целевого адреса простыми (обычно арифметическими) инструкциями из макроса nop. Эти инструкции симулятор выполняет в режиме условного выполнения, в котором результат любой операции объявляется неизвестным. При прогоне теста любое значение условия перехода приведет в целевом адресе к такому состоянию системы, которое не противоречит полученному в симуляторе.

Наконец, возможны безвыходные ситуации, когда неизвестен адрес памяти данных при сохранении или целевой адрес перехода. Генератор должен предотвращать такие ситуации, поскольку они приводят к непредсказуемому состоянию. Если инструкция выполняется впервые, ее можно заменить на nop, а если не впервые (в цикле или в процедуре), то генерация завершается аварийно.

Встроенный симулятор может быть реализован как покомандный либо как потактовый. Потактовое выполнение может быть использовано в генераторе случайных тестов при тестировании конвейера, когда случайный выбор регистров определяется их позициями в конвейере. Однако сложность современного конвейера не соответствует простой модели выбора в терминах "возраста использования" и на основе небольшого набора вероятностей. Для тестирования конвейера можно использовать описанные выше инструменты, такие как переменные и функции. В остальном для

генератора тестов достаточно иметь покомандный симулятор.

Преимущества использования встроенного симулятора следующие:

- Моделирование выполнения увеличивает долю регистров с известным содержимым. Это часто позволяет обойтись без перезагрузки регистров адреса, а также использовать арифметические инструкции для их модификации.
- Отслеживание всех итераций циклов и вызовов процедур упрощает контроль за состоянием целевого устройства и корректностью теста.
- Сбор и сравнение данных при генерации и при прогоне теста позволяет создавать тесты с самопроверкой.

Использование симулятора может также вызывать проблемы, связанные с неизвестными адресами в циклах и процедурах. Однако при генерации случайного кода принимаются все меры, чтобы адреса были корректными. При этом ошибка может возникнуть только за счет заданных инструкций и аргументов, но и тогда организовать ее непросто. В текущей реализации генератора тестов симулятор начинает работу автоматически в начале выполнения программы построения теста в шаблоне, то есть после вызова служебных макросов (start, call, nop, bnt) и формирования исходных данных и служебной процедуры инициализации. Перед вызовом служебного макроса final симулятор снова отключается. Это связано с тем, что дополнительный контроль корректности со стороны симулятора может помешать вызову служебных макросов, при котором достаточно контроля от генератора тестов. Содержимое регистров при вызове служебных макросов неизвестно.Основные проверки корректности кода выполняются при генерации очередной инструкции. Чтобы исключить некоторые (не все) проверки, предоставлены две возможности: оператор Code, который задает готовый код операции, и ассемблерный формат инструкции, в котором генератор игнорирует содержимое регистров и проверяет только основные условия - четность, тип регистров и представимость непосредственных аргументов.

Дополнительная проверка делается для инструкции возврата из процедуры (jr ra), если симулятор включен. Если адрес возврата неизвестен или указывает на занятое место, инструкция отменяется. Эта проверка нужна, чтобы избежать зацикливания при ошибках оформления процедур с однократным вызовом. Также для борьбы с зацикливанием в генераторе контролируется число повторяющихся переходов по одному и тому же адресу. Встроенный симулятор проверяет инструкцию при каждом ее выполнении, независимо от способа ее добавления. Если возникает условие исключительной ситуации либо неизвестен операнд или слово памяти при загрузке, результат объявляется неизвестным. Ошибкой считается только невозможность выполнения инструкции. Симулятор обнаруживает следующие ошибки:

- Несуществующая инструкция или регистр.
- Недопустимая инструкция в слоте задержки перехода.
- Неизвестно содержимое регистра целевого адреса перехода.
- Неизвестно содержимое регистра базы или индекса при сохранении. При загрузке игнорируется.
- Адрес данных не выровнен. При загрузке предупреждение и неизвестное содержимое приемника. При сохранении - ошибка.
- Адрес не принадлежит ни одной из описанных областей памяти. При загрузке предупреждение и неизвестное содержимое приемника. Для pref(x) и cache предупреждение. При сохранении - ошибка.

Можно заметить, что несуществующая инструкция все же может быть добавлена оператором Code в обход симулятора, например - в служебном макросе final или start. Встроенный симулятор можно использовать для создания тестов с самопроверкой. Обычно проверка правильности выполнения делается с помощью запуска теста на двух разных платформах и сравнения протоколов выполнения. Симулятор позволяет упростить проверку. Для этого генератор тестов должен периодически добавлять в код сбор и сравнение данных от встроенного симулятора и данных, полученных при прогоне теста.

6. Заключение

Использование моделей устройств, таких как кэш и конвейер, позволяет повысить содержательность случайных тестов при тестировании этих устройств. Более универсальным способом является построение случайных тестов на основе детерминированных. Язык описания программы предлагает для этого средства - переменные и встроенные функции. Функции обеспечивают случайный выбор и нестандартные действия, а переменные служат для передачи результатов этих действий в тестовый код. При этом не снижается ни степень случайности, ни объем тестов. Проведенные численные эксперименты показали увеличение в тестах доли целевых инструкций.

От встроенного симулятора в генераторе случайных тестов требуются специфические свойства: выполнение кода по мере его генерации, замена ошибочной инструкции, отложенное выполнение. Реализация симулятора в рамках подхода ограниченного моделирования позволяет повысить надежность отбора тестов и гибкость программирования шаблонов. Далее планируется использовать симулятор для создания тестов с самопроверкой.

Improvement of random test generator tergen

A.S. Koutsaev

Abstract: The quality of random tests depends on the relation between test cases and options of random choice. The application of models of devices makes this relation more explicit. Also enhanced random tests may be obtained on the base of direct tests complemented with random choice and usage of tools like variables and built-in functions. In particular, these tools allow to obtain intense random tests for subroutine calls. The built-in simulator also extends the possibilities for testing. It facilitates the checkout of test correctness, makes possible to manage the content of registers and memory by the familiar programmer's tools and allows for creating random tests with internal checkup.

Keywords: random tests, template, simulator, variables, subroutines.

Литература

1. И.Ш. Хисамбеев. Роль стохастического тестирования в функциональной верификации микропроцессоров. Программные продукты и системы 2012, №3, с. 107-112.
2. И.В. Грибков, А.В. Захаров и др. Стохастическое тестирование в системе INTEG. Программные продукты и системы. 2007, № 2, с. 22–26.
3. И.В. Грибков, А.В. Захаров и др. Развитие системы стохастического тестирования INTEG. Программные продукты и системы 2010, №2, с. 14-22.
4. А.С. Куцаев. Генератор стохастических тестов с ограниченным моделированием выполнения. Труды НИИСИ РАН, 2016, Т.6 № 1, с. 69-77.
5. А.А. Лесных, И.А.Широков. Покомандная модель проектируемого 64-битного микропроцессора. В сб. "Методы встречной оптимизации в задачах обработки сигналов". М., НИИСИ РАН, 2005. с. 96-116.

Расчет задержек в межсоединениях цифровых БИС с учётом электротепловых эффектов

К.О. Петросянц[1] , Е.И. Батаруева[2] , Н.И. Рябов[3]

Национальный исследовательский университет «Высшая школа экономики»
(Московский институт электроники и математики им. А.Н. Тихонова), Москва, Россия,

E-mail's: [1] kpetrosyants@hse.ru, [2] yelg@yandex.ru, [3] nryabov1954@gmail.com

Аннотация - В работе ставится задача создания программного обеспечения для моделирования задержек в межсоединениях СБИС с учётом температурных эффектов. Используется модель межсоединения в виде распределённой RC-цепи, параметры которой зависят от распределения температуры на поверхности кристалла, которое рассчитывается с помощью программы «Перегрев – МС». Исходя из распределения температуры вдоль межсоединения, рассчитываются параметры его модели – сопротивления и ёмкости звеньев RC цепи, что позволяет учитывать влияние неравномерного разогрева кристалла на электрические свойства межсоединения. Для упрощения модели межсоединений и уменьшения времени счета многозвенная RC-модель редуцирована в компактную П-образную эквивалентную схему с температурно-зависимыми параметрами. Показано, что погрешность по амплитуде сигнала при переходе к П-образной схеме составляет не более 7%, по фазе 2%. При этом время счета сокращается на 25-30%.

Ключевые слова - межсоединения БИС, редукция, электротепловая модель, задержка сигналов в межсоединениях.

1. Введение

Неравномерный нагрев линий межсоединений приводит к искажению передаваемых сигналов, в частности, увеличению времени задержки, что может вызвать рассогласование сигналов во времени. Распределение температуры в линии межсоединения обусловлено как неравномерным нагревом кристалла, по которому проходит линия, так и током, протекающим в линии (так называемый «саморазогрев»). Кроме того, ток линии вызывает падение напряжения вдоль нее.

В работе ставится задача создания программного обеспечения для моделирования задержек в межсоединениях СБИС с учётом температурных эффектов.

2. Электро-тепловое моделирование межсоединений.

В настоящей работе авторы используют модель межсоединения (рис. 1) в виде распределённой RC-цепи [1], параметры которой зависят от температуры данной точки межсоединения [2,3] (рис. 2).

Распределение температуры на поверхности кристалла рассчитывается с помощью программы «Перегрев – МС» [4]. Исходя из распределения температуры вдоль линии межсоединения, рассчитываются параметры модели межсоединения – сопротивления R(Ti) и ёмкости C(Ti) звеньев RC цепи.

Авторами разработана программа расчёта параметров модели, выводящая результаты расчёта в виде описания электрической схемы в формате SPICE. Такой подход позволяет учитывать влияние неравномерного разогрева кристалла на электрические свойства линии межсоединения.

Рис. 1. Линия межсоединения, проходящая по подложке, отделенная слоем изоляции.

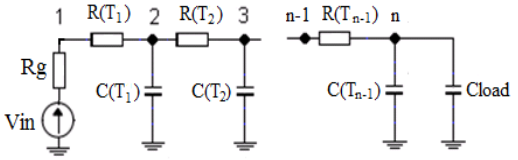

Рис. 2. Модель межсоединения в виде распределённой RC цепи. Величины сопротивлений R и емкостей C зависят от температуры данной точки межсоединения Ti. Vin – напряжение входного генератора, Rg - сопротивление генератора, Cload – ёмкость нагрузки.

С целью упрощения ЭТ-модели межсоединений и уменьшения времени счета многозвенная RC-модель редуцирована в компактную П-образную эквивалентную схему с температурно-зависимыми параметрами (Рис.3). Для этого была использована программа промышленного назначения PRIMA [5,6], основанная на теоретических связях между блок-алгоритмами Арнольди, матрицами Ланцоша и системными матрицами PRIMA.

Точность расчётов редуцированной данным методом цепи и цепи 10-звенной (взятой за точную модель [4]) составляет 87-89%.

Но, несмотря на высокую точность, полученная модель не позволяет учитывать зависимость свойств межсоединения от распределения температуры вдоль него. Поэтому был разработан метод, который подразумевает замену конденсаторов на емкостные резисторы и расчет суммарного сопротивления многозвенной цепи, представляющего из себя сумму сопротивлений одного резистора и одной емкости. Были выведены формулы для расчёта значений параметров П-образной однозвенной модели по параметрам многозвенной RC-цепи, к которой применяется редукция.

Элементы редуцированной RC-цепи имеют множители

$$1+\alpha'\cdot\Delta T,\ \ 1+\theta'\cdot\Delta T, \qquad (1)$$

где α',θ' - температурные коэффициенты сопротивления и диэлектрической постоянной, используемые в П-образной однозвенной модели.

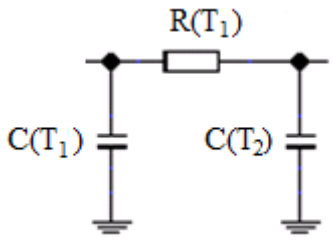

Рис. 3. П-образная эквивалентная схема межсоединения с температурно-зависимыми параметрами

Для определения значений α' и θ' были рассчитаны значения сопротивления *R* и емкости *C* в диапазоне температур от 20°C до 120°C. Таким образом, температурные коэффициенты для нашей П-образной модели межсоединения можно вычислить следующим образом:

$$\alpha' = \frac{R_1 - R_2}{R_1 \Delta T}, \quad \theta' = \frac{C_1 - C_2}{C_1 \Delta T}, \qquad (2)$$

где $\Delta T=(20^0\text{C}-T)$.

Анализ полученных результатов показывает, что приведенные температурные коэффициенты редуцированной цепи зависят от разности температур и могут быть представлены следующими функциями:

$$\alpha' = \alpha \frac{k}{\Delta T}, \quad \theta' = \theta \frac{m}{\Delta T}, \qquad (3)$$

где: α,θ - температурные коэффициенты сопротивления и диэлектрической постоянной материалов межсоединения, k=-120.97, m=684.7.

С учетом выше описанных исследований были разработаны электротепловые SPICE модели, эквивалентные межсоединениям, которые в дальнейшем используются для высокоточного моделирования логических вентилей на разных температурах.

На примере десятизвенной RC-цепи показано, что погрешность по амплитуде сигнала при переходе к П-образной схеме составляет не более 7%, по фазе 2%. При этом время счета сокращается на 25-30%.

3. Расчет задержек межсоединений цифровых БИС

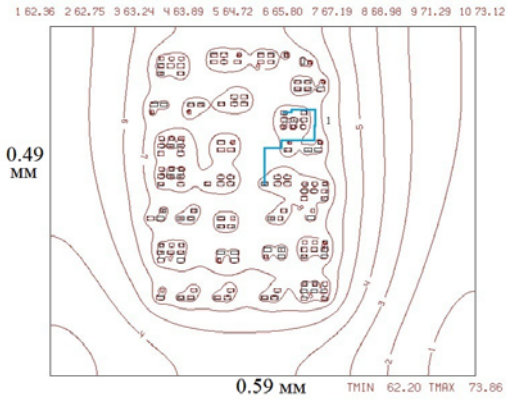

Рис. 4. Топология полусумматора на два входа, входящего в состав БМК серии 6501ХМ1. 1 - линия межсоединения; изолинии температуры поверхности - в градусах Цельсия, соответствие номеров и температур – в верхней строке рисунка.

С использованием разработанных моделей проведен расчет задержек в межсоединениях для микросхемы 4-х разрядного сумматора, реализованного на базе топологических ячеек НЕ, 2ИЛИ-НЕ, 3ИЛИ-НЕ, 4ИЛИ-НЕ, полусумматора на два

входа из состава БМК серии 6501XM1. Схема полусумматора рассеивает мощность 0,37 Вт, занимает на кристалле площадь 0,6•0,5 мм^2 и входит в качестве фрагмента в состав более сложных блоков процессора специализированной ЭВМ со сверхвысоким быстродействием. Для данного фрагмента было смоделировано общее тепловое поле. Полученное распределение температуры вдоль линии межсоединения использовано для расчета параметров ее ЭТ-модели.

Задержка распространения сигнала по линии межсоединения без учета неоднородного температурного профиля составила 22,9 пс, а с учетом 31,9 пс, т.е. пренебрежение тепловыми эффектами вызывает погрешность 28%.

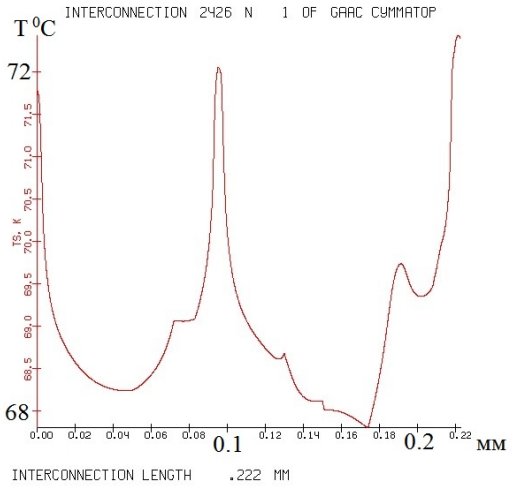

Рис. 5. Распределение температуры кристалла вдоль линии межсоединения.

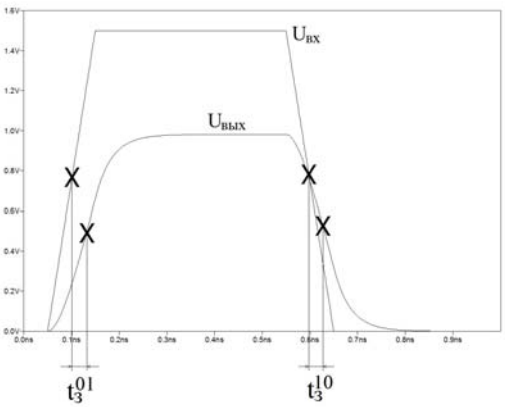

Рис. 6. Результат моделирования задержек линии межсоединения с помощью SPICE. Задержка переднего фронта составляет 31.9 пс, заднего – 31.1 пс.

Кроме того, была проведена оценка совместного влияния электрических и тепловых эффектов на амплитуду напряжения сигнала при его передаче по межсоединению. При входном напряжении 1,5 В в начальной точке металлизированной линии, на ее выходе в результате расчета было получено значение 0,981 В. То есть, потери по амплитуде сигнала составили 35%. Таким образом, в линиях межсоединений наблюдается эффект «просадки» напряжения, который важно учитывать при конструировании БИС.

4. Выводы

Разработано программное обеспечение для анализа задержек и электрических потерь в межсоединениях СБИС в зависимости от температурных эффектов.

Температура линии межсоединения определяется по тепловым полям микросхемы, которые рассчитываются с помощью программы «Перегрев – МС».

Разработана программа расчёта параметров модели межсоединения (сопротивлений и емкостей) в зависимости от температуры в точках межсоединения. Разработана процедура редукции многозвенной распределенной электротепловой цепи в компактную П-образную эквивалентную схему, которая включена в библиотеку моделей программ H-SPICE, LTSPICE и др.

Приведены примеры ЭТ-расчета временных задержек и электротепловых потерь в междсоединениях в цифровых фрагментов БИС.

В отличие от описанных ранее аналитических методов, методика численного расчета задержек в межсоединениях БИС, предложенная в данной работе, позволяет: 1) учитывать произвольное распределение температуры в п/п кристаллах БИС и металлизированных межсоединениях; 2) автоматизировать процедуру синтеза компактной П-образной модели междсоединений БИС с температурно-зависимыми параметрами для расчета цифровых и аналоговых узлов БИС с помощью SPICE подобных программ.

The delay calculation in digital LSI interconnections with account for electro-thermal effects

K.O. Petrosyants, E.I. Batarueva, N.I. Ryabov

Abstract : General purpose of this work is development of program tools for delay modeling in LSI interconnections with account for thermal effects. Authors use the interconnection model in the form of distributed RC-circuit, which parameters depends on the chip surface temperature distribution. The chip surface temperature is calculated by program tool “Overheat-MC”. Interconnection model parameters – resistances and capacitances of RC circuit sections, are calculated on the basis of temperature distribution along the interconnection. This approach allows to take into account the influence of chip non-uniform overheat on interconnection electrical characteristics. For the simplification of interconnection model and CPU time decreasing the multi-sectional RC - model reduced to compact Pi-shaped equivalent circuit with temperature-depended parameters. It is shown that in conversion to Pi-shaped circuit, the signal magnitude error is at most 7%, the signal phase error is 2%. In this case CPU time decreases on 25-30%.

Keywords: LSI interconnections, reduction, electro-thermal model, signal delay in interconnections.

Литература

1. Amir H. Ajami, Member, Kaustav Banerjee, Senior Member, and Massoud Pedram, Fellow. Modeling and Analysis of Nonuniform Substrate Temperature Effects on Global ULSI Interconnects. IEEE Trans. on computer-aided design of intergrated circ. and sys., V. 24 (2005), NO. 6, 849-860.
2. S. Rzepka, K. Banerjee. 1998.Characterization of self-heating in advanced VLSI interconnect lines based on thermal finite element simulation. IEEE Trans. Compon., Packag., Manufac. Techn.-A, vol. 21, no. 3, pp. 406–411.
3. J. Cong and K. S. Leung. Optimal wiresizing under the distributed Elmore delay model. In ICCAD, 1993.
4. К.О.Петросянц, Н.И.Рябов. Программа для ЭВМ «Перегрев МС». Свидетельство № 2007613306 от 6.08.2007 г. об официальное регистрации программы для ЭВМ.
5. Altan Odabasioglu, Mustafa Celik. PRIMA: Passive reduced-order interconnect macromodeling algorithm. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 17(8) (1998), 645-654.
6. М.М.Гурарий , М.М.Жаров, С.Г.Русаков, С.Л.Ульянов. Методы возмущений и селективные методы в задачах редукции высокоразмерных моделей. Проблемы разработки перспективных микро- и наноэлектронных систем. Сборник научных трудов / под общ. ред. А.Л.Стемпковского. М.:ИППМ РАН, (2008), с. 86-91.

Свойства тока Ion двух затворных КНИ КМОП нанотранзисторов с ассиметрично-легированной рабочей областью

Н. В. Масальский

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: volkov@niisi.ras.ru

Аннотация: Численно исследуется поведение важного параметра КМОП транзистора – тока Ion применительно к ассиметрично-легированным двух затворным КНИ КМОП нанотранзисторам. Анализируется вариант ассиметричного канала (считая от истока): высоколегированная и низколегированная области. Результаты модельных расчетов распределения потенциала суб-50 нм структур находятся в хорошем соответствии с данными моделирования, полученными при помощи коммерчески доступного программного пакета ATLAS^{TM}. На основании полученных распределений потенциала вычисляются вольт-амперные характеристики при помощи сформулированной в рамках зарядового разделения токовой модели. Для выбранных топологических норм оптимизация параметров ассиметричного профиля легирования предоставляет дополнительную возможность управления уровнем тока Ion.

Ключевые слова: двух затворный КНИ КМОП нанотранзистор, неравномерно-легированная рабочая область, распределение потенциала, вольт-амперные характеристики, ток Ion.

1. Введение

Современные тенденции развития КНИ КМОП транзисторных архитектур определили перспективное направление - неравномерно-легированная архитектура - некая обобщенная конструкция, чтобы удовлетворить всевозрастающие требования к элементной базе СБИС [1]. В ближайшем будущем данная архитектура может стать индустриальным стандартом. Применение ассиметрично-легированной рабочей области в КНИ КМОП транзисторе позволяет улучшить его статические и динамические характеристики [1-3]. Например, контролировать напряжения его пробоя за счет ограничения электрического поля в рабочей области транзистора. Топологические параметры конструкции рабочей области оказывают влияние на вольт-амперные характеристики КНИ КМОП нанотранзисторов, что особенно важно при проектировании СБИС с учетом вариаций технологического процесса.

В настоящей работе для двух затворного КНИ КМОП нанотранзистора анализируется вариант ассиметрично-легированной рабочей областью (считая от истока): высоколегированная и низколегированная области с разными протяженностями. Структура рассматриваемого транзистора представлена на рис. 1. Здесь выделим следующие компоненты: 1 – ассиметрично-легированная рабочая область с общей протяженностью L и толщиной t_S, 2 – область истока, 3 область стока, 4 – подзатворный диэлектрик с толщиной t_{ox} для верхнего и нижнего затворов.

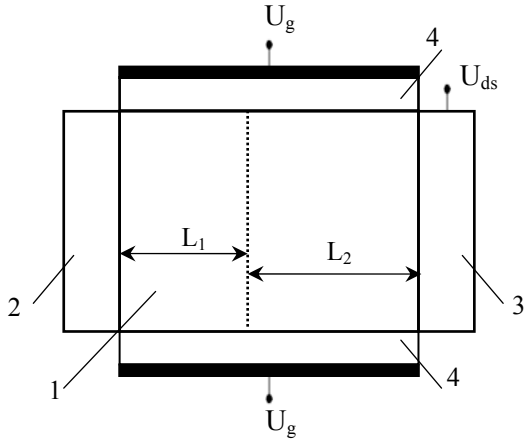

Рис. 1. Структурная схема, где Ug – напряжение на затворах, Uds- напряжение сток/исток (или напряжение питания)

Рабочая область транзистора состоит из двух частей с разными концентрациями легирования так, что выполняются следующие условия:

$$L = L_1 + L_2$$

$$N_i = \begin{cases} N_1, 0 \le y \le L_1, i = 1 \\ N_2, L_1 \le y \le L, i = 2 \\ N_1 > N_2 \end{cases} \quad (1)$$

где N_i – концентрация легирования рабочей области, i = 1 область 1 с протяженностью L_1 и концентрацией легирования N_1 и i = 2

область 2 с протяженностью L_2 и концентрацией легирования N_2.

Целью работы является исследование свойств важного параметра полевого транзистора – тока Ion, для определения условий оптимального выбора величин и отношений L_1 и L_2 и N_1 и N_2 для реализации требуемого уровня тока Ion.

Исследование будет выполнено применительно к прототипу двух затворного симметричного КНИ КМОП транзистора с технологическими параметрами L=45 нм, t_S=10 нм, t_g=1.8 нм, N_1=1.0x10^{17} см$^{-3}$, N_2=1.0x10^{15} см$^{-3}$.

Максимальный уровень легирования стока/истока 1x10^{20} см$^{-3}$.

Моделирование характеристик рассматриваемого нанотранзистора осуществляется при помощи распределения потенциала в рабочей области транзистора.

Данное распределение получено из аналитического решения 2D уравнение Пуассона с расширенными граничными условиями.

В них включены соотношения для учета эффекта асимметрично легированной рабочей области, которые соответствуют случаю однородности потенциального и электрического полей на границе соприкосновения разно легированных частей.

Полученные распределения поверхностного потенциала сопоставлялись с данными коммерчески доступной программы моделирования ATLAS [4].

Далее, при помощи концепции зарядового разделения рассчитывались вольт-амперные характеристики (ВАХ) устройства в режиме сильной инверсии с учетом полевой и концентрационной зависимости подвижности.

Анализируемый подход моделирования позволяет с высокой эффективностью определить характеристики тока Ion в широком диапазоне топологических параметров двух затворных ассиметрично легированных КНИ КМОП нанотранзисторов

1. Результаты моделирования распределения потенциала

На рис. 2 приведены рассчитанные распределения поверхностного потенциала в рабочей области транзистора вдоль затвора для двух отношений $\frac{L_1}{L}$.

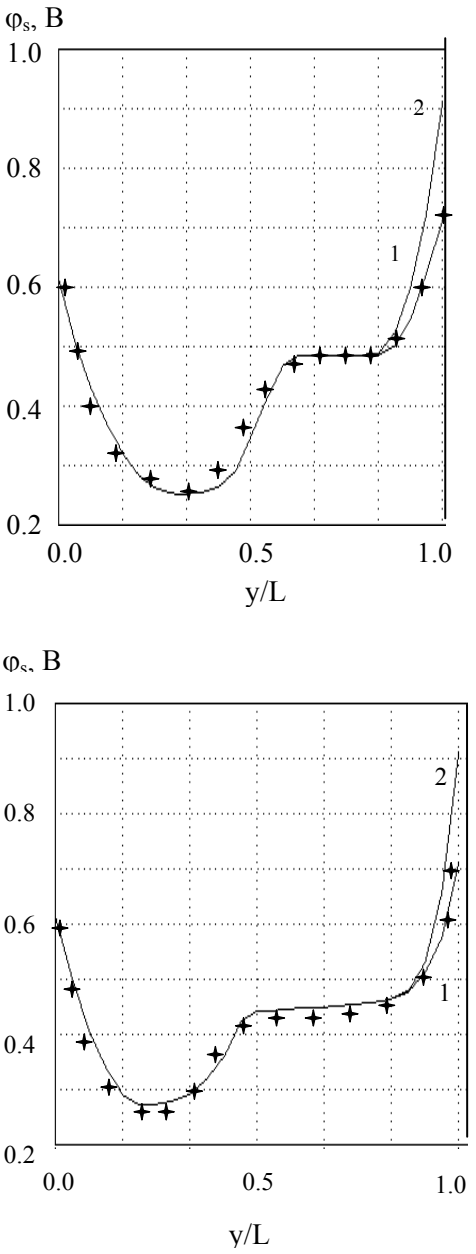

Рис. 2. Распределение поверхностного потенциала вдоль рабочей области, где верхний рисунок $L_1 = L_2$, нижний рисунок $2L_1 = L_2$. Для обоих рисунков: 1- U_{ds} = 0.1 В, 2- U_{ds} = 0.3 В, звездочкой обозначенные даны моделирования, полученные при помощи программы ATLAS

Сравнение результатов расчетов с данными моделирования, полученными при помощи программного пакета ATLAS^{TM} для 2D моделирования транзисторных структур, позволяет сделать вывод о хорошем соответствии между ними.

Из сопоставления распределения потенциала в неравномерно и однородно легированных транзисторных структурах, хорошо видно различие в форме распределения потенциала. Распределение потенциала в однородно легированном

транзисторе при низком напряжении сток/исток Uds характеризуется квази симметричной формой и практически равномерной зависимостью распределения потенциала от данного напряжения. Такое поведение обусловлено линейной зависимостью продольного электрического поля в рабочей области от Uds.

В общем случае рассматривались конфигурации рабочей области в пределах $0 \le \frac{L_1}{L} < 1$, где 0 - случай нелегированной рабочей области, 1 – случай однородно-легированной рабочей области. В общем случае форма распределение потенциала в высоколегированной области практически не зависит от напряжения Uds. Небольшое влияние проявляется на границе двух разно-легированных областей. Минимум поверхностного потенциала расположен в высоколегированной области, ближе к середине между центром рабочей области и центром высоко легированной области. Однако это правило не применимо для тонких (протяженностью менее 0.1L) высоколегированных областей

Главная особенность в распределении потенциала неравномерно легированных структур сконцентрирована в низколегированной области. В окрестности стыковки двух областей потенциал испытывает резкий скачок. При этом во всей низколегированной области потенциал практически остается неизменным. И ближе к стоку он начинает возрастать, но с меньшим градиентом, чем потенциал в однородно легированной структуре. Такая ассиметричная форма распределения потенциала способствует улучшению функциональных характеристик транзистора, в частности оказывает влияние на распределение продольного электрического поля [5, 6].

Распределение потенциала определяется также отношением концентраций N_1 и N_2. При снижении параметра N_2 минимум потенциала сдвигается к истоку. Разница между минимальным значением потенциала и потенциалом соответствующим пологой части возрастает. При возрастании N_2 пологая часть в распределении потенциала постепенно исчезает, а минимум потенциала сдвигается к истоку. При выравнивании концентраций распределение потенциала в точности соответствует распределению потенциала для однородно-легированного случая.

В распределении электрического поля в рабочей области ассиметрично-легированном и равномерно-легированном транзисторах будут проявляться существенные различия. Для равномерно-легированного транзистора, распределение электрического поля носит в целом монотонный характер и характеризуется только резким скачком у стока. Для ассиметрично-легированного транзистора распределение электрического поля существенно иное. Основной рост сосредоточен в высоколегированной области. Практически во всей низколегированной области электрическое поле меняется незначительно и только у стока оно резко возрастает. Такое поведение электрического поля в рабочей области транзистора приводит к ослаблению эффектов, связанных с зарядовым разделением. Это является одним из основных преимуществ ассиметрично-легированной архитектуры. Снижение величины электрического поля у стока для неравномерно легированной структуры составляет приблизительно 20% по сравнению с однородно легированным устройством.

3. Моделирование вольт-амперных характеристик

Для расчета ВАХ в режиме сильной инверсии сформулирована аналитическая модель в рамках концепции зарядового разделения с учетом свойств высоколегированной и низколегированной частями кремниевой рабочей области транзистора. Для каждой части выражение для тока получено по апробированной методике [7], исходя из плотности заряда с учетом модифицированного выражения для скорости насыщения, моделью подвижности Lombardi с учетом высокой полевой деградации, эффектов легирования и длины разнородно легированных областей. В модели предполагается наличие крутого перехода концентраций в кремниевой рабочей области на границе высоко- и низколегированных зон. Это означает, что между ними отсутствует боковая диффузия. Обобщенное выражение тока транзистора получено суммированием парциальных токов по всей рабочей области.

На рисунке 3 приведена экстрагированная из расчетов зависимость тока Ion от значения параметра L_1 / L.

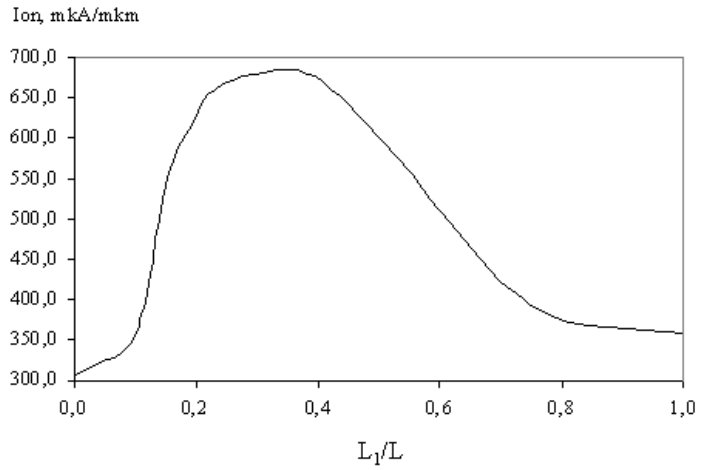

Рис. 3. Зависимость Ion от отношения L_1/L при Uds=Ug=0.8 В

Из приведенных данных следует, что самый высокий ток 679 мкА/мкм соответствует значению параметра $L_1 / L =$ 0.38, что в физическом выражении составляет 17.1 нм. В диапазоне значений 0.14…0.8 все асимметрично легированные конфигурации имеют более высокие значения Ion по сравнению с аналогичным однородно и нелегированным прототипами. В области больших значений L_1 / L>0.8 максимальное значение тока практически не изменяется из-за отсутствия экранировки стока, что способствует проявлению деградационных эффектов, свойственных равномерно легированным транзисторным структурам [8, 9]. В области малых значений L_1 / L<0.1 величина ток Ion сопоставима с током для нелегированного случая из-за очень узкой высоколегированной области.

На рисунке 4 приведена экстрагированная из расчетов зависимость тока Ion от уровня концентрации N_1 при $L_1 / L =$ 0.38. Из представленных результатов видно, зависимость Ion(N_1) близка к линейной.

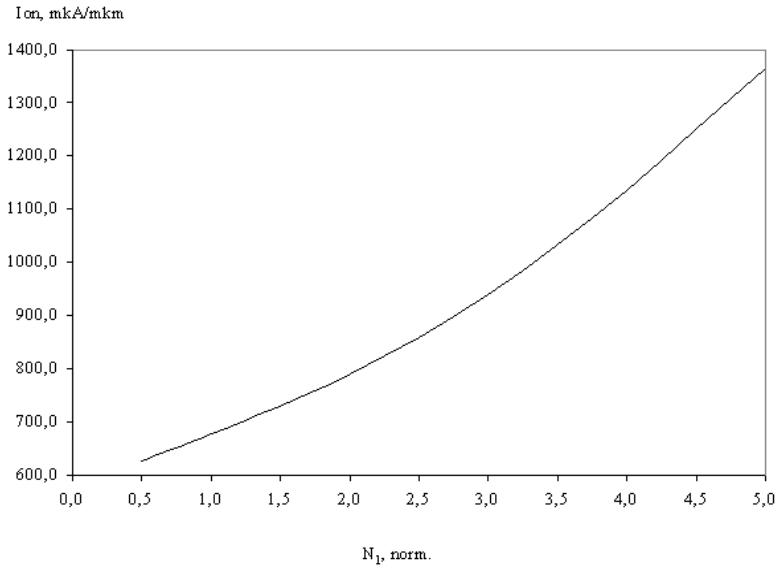

Рис. 4. Зависимость Ion от нормированного N1 ($x10^{17}$) при L_1/L=0.38 и Uds=Ug=0.8 В

Следует отметить, что по сравнению с одно затворным КНИ КМОП транзистором ток насыщения анализируемых конфигураций будет выше при прочих равных условиях [6].

Из результатов численных экспериментов следует, что для широкого диапазона значений параметра L_1 / L поведение тока насыщения обуславливает существенное уменьшение выходной

проводимости g_d, которое приводит к росту напряжения Эрли U_E [5]. Такие значения U_E и g_d достигаются из-за наличия низколегированной зоны в рабочей области, маскирующей понижение напряжения стока, что сохраняет неизменной электронную концентрацию в инверсионном слое. Данные прототипы могут найти применение в проектах устройств с низкой мощности и низковольтных приложений аналоговых схем [10, 12].

Следует отметить, что расчеты, проведенные для субмикронных структур, показали хорошее соответствие между аналитическим и численным моделированиями, но также и между этими моделированиями и экспериментальными значениями [6, 10-12].

Заключение

По результатам численного моделирования показано, что двух затворные симметричные КНИ КМОП нанотранзисторы с ассиметрично-легированной рабочей областью достигают более высоких значений тока Ion по сравнению с традиционными КНИ КМОП транзисторами, даже когда длина затвора становится меньше, чем 50 нм. Варьирование протяженностями разно легированных областей и их концентрациями предоставляет дополнительную степень свободы для оптимизации значений электро-физических параметров дух затворных ассиметрично-легированных КНИ КМОП нанотранзисторов, в частности, тока Ion. В целом, все улучшения в характеристиках транзисторов являются следствием самой ассиметричной архитектуры с комбинацией высоко- и низко-легируемой областей.

Представленный подход моделирования позволяет для заданных топологических норм с высокой эффективностью определить условия оптимального выбора топологических параметров двух затворных КНИ КМОП нанотранзисторов с ассиметрично-легированной рабочей областью.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) “43.35. Развитие аналитических методов моделирования квантово-размерных компонентов высоко-производительных СБИС”. 0065-2018-0006).

Properties of current Ion for the double gate SOI CMOS nanotransistors with an asymmetric-doping channel

N. Masalsky

Abstract: Numerically the behavior of the CMOS transistor important parameter – current Ion is researched relation to the double gate SOI CMOS nanotransistors with a asymmetric-doping channel. The version of the asymmetric channel is analyzed (including from a source): high-doping and low-doping areas. Results of numerical calculations of potential distribution of sub-50 nanometer structures are in good compliance of simulation data, received by means of commercially available software package ATLAS^{TM}. Based on the received potential distributions volt ampere characteristics by means of the current model formulated within charge division are calculated. For the selected topological norms optimization of parameters of an asymmetric doping profile gives an additional opportunity for control of Ion current layer.

Keywords: double gate SOI CMOS nanotransistor, uniform-doping channel, potential distribution, volt-ampere characteristics, current Ion

Литература

1. Н.В. Масальский. Двух затворные неравномерно легированные полевые нанотранзисторы. Saarbrücken, Germany, LAP LAMBERT Academic Publishing GmbH & Co, 2016.
2. D.A. Neamen. Semiconductor physics & devices: basic principles. New York, McGraw-Hill, 2011.

3. Н.В. Масальский. Моделирование характеристик логических вентилей на симметричных двух затворных КНИ КМОП нанотранзисторах с ассимметрично-легированной рабочей областью. «Труды НИИСИ РАН». т. 7(2017), № 2, 132-140.
4. URL: http://www.silvaco.com/ Silvaco Int. 2004: ATLAS User's Manual A 2D numerical device simulator (дата обращения 25.01.2017).
5. M. A. Pavanello, J. A. Martino, V. Dessard, D. Flandre. Analog performance and application of graded-channel fully depleted SOI MOSFETs. «Solid-State Electronics». v. 44(2000), № 4, 1219-1222.
6. J. S. Martin, A. Bournel, P. Dollfus. Comparison of multiple-gate MOSFET architectures using Monte Carlo simulation. «Solid-State Electronics». v. 50(2006), № 1, 94-101.
7. Н.В. Масальский. Полностью обедненные КМОП КНИ логические элементы для низковольтных применений. «Микроэлектроника». т. 37(2008), № 6, 470-480.
8. A. Savio, S. Monfray, C. Charbuillet, T. Skotnicki. On the limitations of silicon for I-MOS integration. «IEEE Transactions on Electron Devices». v. 56(2009), № 5, 1110–1117.
9. H. Wong, Y. Fub, J.J. Liou, Y. Yue. Hot-carrier reliability and breakdown characteristics of multi-finger RF MOS transistors. «Microelectronics Reliability Elsevier Journal». v. 49(2009), № 1, 13–16.
10. M. A. Pavanello, J. A. Martino, J.-P. Raskin, D. Flandre. High performance analog operation of double gate transistors with the graded-channel architecture at low temperatures. «Solid-State Electronics». v. 49(2005), № 6, 1569-1575.
11. D. D. Lu, M. V. Dunga, C-H. Lin, A. M. Niknejad, C. Hu. A computationally efficient compact model for fully-depleted SOI MOSFETs with independently controlled front- and back- gates. «Solid-State Electronics». v. 62(2011), № 1, 31–39.
12. Н.В. Масальский. Характеристики двух затворных КМОП нанотранзисторов с градиентно-легированной рабочей областью. «Труды НИИСИ РАН». т. 6(2016), № 2, 77-82.

Система тестирования логических моделей KMDTESTKIT

С.А.Сидоров[1], **А.Б.Слепов** [2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1] *sidorov@niisi.msk.ru,* [2] *sir-lexa@yandex. ru*

Аннотация: Описана Система тестирования KMDTESTKIT для программной верификации логических моделей систем на кристалле. Изложены основные задачи, решаемые системой и способы решения, ее состав и структура.

Ключевые слова: программное тестирование, верификация, система на кристалле

Введение

Маршрут разработки микросхем от замысла (технического задания) до серийного выпуска включает множество этапов, на каждом из которых требуется верификация проекта. Такой маршрут, применяемый в ФГУ ФНЦ НИИСИ РАН, подробно описан в [1, 2, 3, 4].

В этих работах показано место и значимость процедур верификации проекта на всех стадиях разработки и производства, описаны виды и методы тестирования микросхем. Эта задача не нова, разработчики применяют различные методики и подходы [5, 6, 7, 8]. Однако общепринятого инструмента до сих пор нет.

Для устройств с микропроцессором, систем на кристалле (СнК), на первый план выходит методика верификации программными тестами.

Программное тестирование СнК и отдельных микропроцессоров применимо и необходимо на всех этапах разработки: поведенческое моделирование, разработка RTL, NetList, отработка на ПЛИС, отбраковка микросхем на производстве и проверка в составе изделия – электронного модуля или ЭВМ.

Каждый этап диктует свои условия и требования к характеристикам тестов – время исполнения, полнота тестирования, объем кода, степень информативности диагностических сообщений и пр.

Так, специфика верификации на моделях и проверки на производственном оборудовании накладывает существенные ограничения по времени тестирования и в меньшей степени по объему программного кода, что приводит к появлению «ниши» программных тестов нижнего уровня, удовлетворяющих требованиям по времени работы.

Обычно это максимум десятки миллисекунд модельного времени, что соответствует от нескольких часов (на RTL) до нескольких дней (на NetList) времени моделирования.

Эти тесты представляют собой часть общего тестового набора изделия, включающего еще комплексные тесты, нагрузочные тесты, запуск операционных систем как метод тестирования, а также применение обширного тестового арсенала, разработанного мировым сообществом, например, для ОС Linux.

Хотя программные тесты нижнего уровня кажутся лишь небольшой частью тестового набора, они представляют доминирующий инструмент функциональной верификации на этапе разработки микросхемы. Они и наиболее сложны в разработке, поскольку требуют от тестировщика детального знания проверяемой аппаратуры, причем не только программной модели, но и структуры этой аппаратуры, протоколов физического уровня и пр. Поэтому одной из главных задач при разработке программных тестов нижнего уровня становится обеспечение возможности повторного использования тестов как на разных этапах разработки микросхемы, так и в новых проектах.

В ФГУ ФНЦ НИИСИ РАН в разработке всегда находится несколько проектов на разных стадиях, при этом новые проекты базируются, естественно, на результатах завершенных работ. Завершенные работы, в свою очередь, необходимо сопровождать в течение всего жизненного цикла изделий, т.е. не один десяток лет. В каждом проекте требуются сотни тестов, таким образом, общее количество тестов, с которыми ведется работа, активно или изредка, становится необозримым. При этом заметная часть этой массы является, по сути, вариантами одного и того же теста.

Для решения задачи организации программного тестирования низкого уровня потребовался инструмент, позволяющий справиться с большим количеством тестов и решающий следующие задачи:

- накопление банка тестов,
- обеспечение тестирования на разных этапах,
- унификация тестов,

- обслуживание нескольких проектов-платформ,
- сохранение истории проектов,
- поддержка коллективной разработки,
- использование нескольких логических моделей (подсистем),
- регрессионное тестирование, в т.ч. на модулях,
- случайное тестирование.

Таким инструментом стала разработанная в НИИСИ РАН система KMDTESTKIT, или сокращенно KMD. Не претендуя на статус универсального инструмента, KMD создана и развивается для максимального удовлетворения потребностей принятого в НИИСИ маршрута разработки и является его неотъемлемой частью.

Рассмотрим более подробно базовые понятия, решаемые задачи, состав и устройство KMDTESTKIT.

Базовые понятия

Платформа (platform) – проект конкретной микросхемы, оформляемый в KMD отдельной директорией с описаниями ее программной модели, библиотеками, скриптами и прочими индивидуальными файлами и настройками.

Тест – директория, содержащая исходные тексты тестовой программы, Makefile, дополнительные файлы при необходимости. В подавляющем большинстве случаев исполняемые файлы теста не хранятся, а создаются всякий раз при запуске.

Пакет тестов (testpack) – директория с тестами. Тесты в пакет подбираются по принадлежности к устройству (например, CAN или I2C) или к платформе.

Регрессионное тестирование, регрессия – выполнение всего имеющегося набора тестов для данной платформы для проверки ее работоспособности. Как правило, выполняется после модификации моделей.

Основные задачи KMD и как они решаются

Фундаментальная задача – **накопление банка тестов**, т.е. сохранение и приумножение плодов нелегкого труда тестировщиков за многие годы. Очевидно, простое складирование в данном случае не подходит, поскольку тест привязан к определенному проекту и, возможно, конкретной его версии, компилятору и системе моделирования, которые обновляются чаще, чем ежегодно. Хранилище необходимо структурировать таким образом, чтобы для каждого проекта было описано однозначно его подмножество тестов, имелась возможность проверить их работоспособность.

Единицей структурирования выбран пакет тестов (testpack). Пакеты систематизированы либо по устройствам, например, SPI, FPU в составе процессора, или МКИО; либо по проектам – платформам. В пакете собраны тесты, относящиеся, соответственно, либо к устройству, либо к данной микросхеме. В каждом пакете имеется перечень тестов с комментариями. Этот перечень используется при выборе тестов для регрессионного тестирования, там же помечаются исключения тестов для данной платформы или режима тестирования.

Такая организация вполне обеспечивает требуемый порядок и позволяет достаточно просто и гибко использовать тестовые пакеты.

Возможность применения одних и тех же тестов **на различных этапах разработки** СнК обусловлена, в первую очередь, параметризацией тестов. Так, например, для RTL- и NetList-моделей проверка “накристальной” памяти большого объема целиком именно как памяти нецелесообразна (слишком долго), гораздо важнее проверить все линии адреса и данных, возможность обращения различными форматами и пр. При переходе к натурным исследованиям – отбраковка на пластине, проверка микросхемы в составе электронного модуля, - необходимо проверять весь доступный объем памяти, сохраняя разнообразие проверок. Время тестирования в этом случае перестает быть настолько критическим фактором.

Еще один аспект состоит в создании автономных тестов. KMD подразумевает, что в системном ПЗУ имеется стартовый код - boot, включающий инициализацию аппаратуры и первичную обработку прерываний. В модели после инициализации boot передает управление собственно тесту в ОЗУ и их дальнейшее взаимодействие осуществляется при обработке прерываний. Для запуска теста на электронном модуле такая схема не годится, поскольку модульный boot, именуемый Программой ПЗУ, может не совпадать по тем или иным параметрам с ожиданиями тестовой программы. Автономная тестовая программа включает в себя все необходимое, чтобы после старта уже не обращаться к Программе ПЗУ, а обрабатывать прерывания самостоятельно. Ее нужно только загрузить в ОЗУ модуля и стартовать. В таком виде тест собирается в KMD при задании параметра (опции, ключа) “auto” при запуске. При этом программа, собранная как автономная, работает и в моделях.

Необходимость **унификации тестов** возникает естественным образом, как только их количество переваливает за десяток. Унификация распространяется как на оформление, так и на концептуальные требования. В KMD

унификация проведена по следующим параметрам.

Самодостаточность, т.е. тест должен самостоятельно проводить все проверки и диагностировать обнаруженные неисправности, не требуя внешних средств (программ, скриптов) для проверки и сравнения результатов.

Восстановление состояния тестируемых и используемых устройств СнК по окончании тестирования, в т.ч. в случае ошибки. Это необходимо для корректной работы последовательности тестов без перезапуска системы. В ряде случаев это невозможно, такие особые тесты помечаются.

Устойчивость к зацикливанию: тест ни в коем случае не должен уходить в бесконечные циклы, все ожидания какого-либо события должны ограничиваться таймаутом.

Тест должен работать как в *кешируемом*, так и в *некешируемом* пространстве. Если невозможно, то это должно явно прописываться в описании теста.

Отсутствие параметров: тест не должен подразумевать передачу ему каких-либо входных параметров.

Неинтерактивность: тест не должен предполагать какой-либо обратной связи от оператора.

Схема запуска одинакова для всех платформ и тестов: вначале работает boot, располагающийся в системном ПЗУ по стандартному для архитектуры адресу 0xBFC0_0000, который инициализирует необходимую аппаратуру и передает управление программе в ОЗУ по типовому адресу 0x8010_0000. Конечно, ряд тестов не вписывается в такую схему, но это прогнозируемые исключения, и их немного.

Унифицированные *описания и библиотеки* включают индивидуальные для каждой платформы, но единообразные заголовочные файлы с базовыми адресами устройств, распределением векторов прерываний и пр., структурами-описателями регистров устройств. Базовые библиотеки могут быть индивидуальными для платформы или поддерживать группу платформ, но названия и форматы функций одинаковы. В базовые библиотеки включены поддержка вывода на консоль, использование таймеров, обработка прерываний, и т.д. Для устройств, требующих множества тестов (например, SpaceWire, CAN), обычно формируются специализированные библиотеки для этих групп тестов.

Диагностика подчиняется нескольким правилам: каждый тест по окончании возвращает параметр, означающий отсутствие ошибок – “0”, или их наличие – ненулевое значение; сообщение об ошибке должно быть кратким (не более 2-3 строк), но информативным (название теста/подтеста, режим тестирования, суть ошибки); для длительных тестов (от десятков минут при моделировании) необходимы промежуточные сообщения о прохождении теста.

Поддержка нескольких проектов-платформ реализована классическим методом выделения платформенно-зависимых компонентов и обобщения независимых.

Для каждой платформы имеются свои заголовочные файлы с распределением адресного пространства и структурами регистровых файлов устройств СнК, библиотеки и Makefile.

Индивидуальны также скрипты, управляющие выполнением тестов – компиляцией, запуском модели, отработкой опций запуска. Остальные ресурсы – общие библиотеки, пакеты тестов, инфраструктура общие. Настройка на конкретную платформу глобальная, т.е. одновременно работать можно только с одной платформой.

Сохранение истории проектов и поддержка коллективной разработки обеспечивается применением системы управления версиями Subversion, иначе называемой SVN [9]. Завершенные, точнее, зафиксированные проекты просто ассоциируются с номером ревизии SVN на момент фиксации, и всегда могут быть восстановлены из репозитория, а поддержка коллективной работы является одной из основных возможностей SVN.

Выбор именно Subversion для этих целей был вполне осознан, исходя из опыта работы, задач и специфики разработки.

Использование нескольких логических моделей (подсистем) для одной платформы необходимо из-за различий в конфигурации тестбенча (testbench) – аппаратного окружения тестируемого устройства.

Например, дискретные сигналы могут использоваться просто как линии ввода-вывода, или для выполнения специальных функций: chip select для устройства, вход или выход частоты таймера и пр. Разные модели также предназначены для уровня RTL и NetList.

Выбор конкретной модели выполняется при запуске моделирования простым заданием ключа.

Регрессионное тестирование, то есть периодическая проверка проекта полным набором тестов, осуществляется двумя способами. Первый – это последовательное выполнение всех тестов с накоплением результатов, выполняется на одном сервере. Второй – использование Sun Grid Engine (SGE) [10], позволяющий запускать тесты на всех доступных серверах параллельно, что существенно сокращает время тестирования – фактически, до времени выполнения самого длинного теста. В KMD поддержаны оба способа, хотя преимущественно используется быстрый.

Оба подхода позволяют выполнять как полную регрессию, так и запуск всех тестов в одном тестовом пакете.

Отдельный механизм предназначен для выполнения регрессии не на модели, а на реальном электронном модуле, для чего разработана подсистема builder.

Она будет описана ниже.

Случайное тестирование, точнее, тестирование с использованием генерации случайных тестовых последовательностей, легко укладывается в механизм работы KMD.

При запуске случайного теста всегда происходит новая генерация, поэтому нет необходимости хранить множество сгенерированных тестов.

Достаточно лишь выполнять такой запуск в цикле, что обеспечивается заданием ключа при запуске.

Генератор случайных тестовых последовательностей [11, 12] выдает программный код для выполнения на модели, а контроль правильности осуществляется либо параллельным исполнением RTL- и поведенческой моделей с покомандным контролем, либо последовательными выполнениями теста на логической и поведенческой моделях с последующим автоматическим сравнением трасс.

Как устроен KMDTESTKIT

KMD организована в виде структуры директорий и хранится в репозитории SVN.

Для работы используется локальная рабочая копия, настроенная на конкретную платформу.

Если в систему вносятся какие-либо изменения или дополнения, то они публикуются средствами SVN.

Файловая структура рабочей копии KMD (* означает название платформы) имеет такой вид:

doc/	Общая документация по KMD
etc/	Файлы, используемые системой сборки – Makefile-ы, скрипты и пр.
include/	Заголовочные файлы общих библиотек, общие заголовочные файлы
lib/	Общие библиотеки
log/	Лог-файлы регрессионного тестирования
platforms/	Платформы
platforms/*/boot/	Программы начальной инициализации для различных режимов тестирования
platforms/*/etc/	Вспомогательные файлы, специфичные для платформы
platforms/*/include/	Заголовочные файлы платформы
platforms/*/lib/	Локальные библиотеки
platforms/*/tools/	Утилиты и настроечные файлы
platforms/*/testpacks	Список тестовых пакетов платформы
testpacks/	Пакеты тестов
testpacks/CAN	Тесты контроллера CAN
testpacks/DEVICES	Тесты устройств
. . .	. . .
testpacks/SYS_UNIT	Тесты контроллера тактовых сигналов
tools/	Общие скрипты и утилиты
testkit.settings	Параметры данной рабочей копии KMD

Для каждой платформы у пользователя создается своя рабочая копия KMD. По сути, различие между рабочими копиями для разных платформ состоит в скомпилированных библиотеках, учитывающих состав устройств и распределение адресов для конкретной платформы.

Для работы с одним тестом используется команда (скрипт) all, выполняемая из директории теста. После команды через пробел можно перечислить требуемые опции. Набор опций варьируется на разных платформах, однако значительная их часть общая. Ниже приведены наиболее часто используемые опции и их краткое описание.

-b - сборка теста (build). Выполняет ‘make clean; make’ и, при необходимости, пред- и пост-действия, такие, как генерация случайных последовательностей или преобразование формата загружаемого файла.

-c - запуск теста на C-модели.

-r - запуск теста на RTL- или NetList-модели.

-t - запуск сравнения трасс выполнения теста на RTL- и C-модели.

-z - запуск теста в циклическом режиме. Обычно используется для случайных тестов, поскольку на каждом шаге цикла заново генерируется тестовая последовательность.

Эти опции (называемые минус-опции) комбинируются, например, ‘-brct’ означает

пересобрать тест, выполнить его на RTL- и C-моделях и сравнить полученные трассы.

Другие опции задаются словами и предписывают варианты сборки теста или моделирования:

help - выдать список доступных опций;

auto - собрать тест для автономной работы т.е. включить в него обработчики прерываний и необходимые настройки, которые находятся в файле boot. Обычно используется для сборки тестов, предназначенных для прогона на аппаратуре, а не на логической модели;

coverage - собрать информацию о тестовом покрытии;

gui - выполнить моделирование в графической оболочке;

intstorm - включить «шторм прерываний», т.е. выполнение теста будет проходить на фоне частых прерываний от таймера;

nocache - отключить кеш-память;

two - моделировать на двухчиповом тестбенче;

и другие.

Имеется также возможность передавать симулятору напрямую любые ключи, не анализируя их в KMD. Это выглядит так:

=<arg>

где <arg> - то, что передается симулятору и должно быть записано в одно слово (без пробелов), состоять из букв, цифр и знаков '_', '-', '+', '='. Например:

=+FREQ_75 =+CDIS =+MCLK=50

Этот набор предписывает установить частоту PLL в 75 МГц, отключить кеш-память и задать опорную частоту 50 МГц.

При запуске теста на RTL-модели в папке теста создаётся папка 'rtlrun<суффикс>', которая содержит все необходимое для запуска RTL-модели, а также получившиеся при запуске RTL-модели трассы (если были созданы). Поле <суффикс> в названии директории формируется из заданных при вызове команды 'all' опций (не начинающихся с минуса), перечисленных через точку в алфавитном порядке. Например: 'rtlrun.auto.intstorm.two'.

Команда (скрипт) 'kmd' предназначена для настройки рабочей копии KMD и работы с тестовыми наборами. Конкретный состав тестов определяется уровнем иерархии, откуда выполняется команда kmd.

Если в директории теста, то действие относится только к этому тесту, из тестового пакета (testpack) – действие распространяется на этот пакет, из корневой директории рабочей копии – ко всем тестовым пакетам, предназначенным для данной платформы.

Формат команды такой:

kmd <действие> [опции]

Список возможных действий можно получить командой 'kmd help'.

Наиболее часто используемые действия:

platform - задать платформу для рабочей копии KMD;

build - пересобрать все тесты из набора;

clean - выполнить 'make clean' для всех тестов;

distclean - выполнить 'make distclean' для всех тестов, т.е. удалить все сгенерированные файлы и директории, в т.ч. собранные библиотеки;

getready - выдать список всех тестов для данной платформы;

regression - выполнить регрессионное тестирование последовательным запуском всех тестов;

и другие.

Регрессионное тестирование

Для регрессионного тестирования в KMD существует также возможность запуска тестов в пакетном режиме на Sun Grid Engine (SGE) [10]. Для этого используется команда 'kmd qrun', принимающая те же самые аргументы регрессии (опции), как и скрипт 'all'.

С помощью 'kmd qrun' можно как запустить регрессионное тестирование на SGE, так и организовать запуск случайных тестов в циклическом режиме (опция '-z'). При выполнении команды 'kmd qrun' формируется список подходящих тестов, и задачи на выполнение этих тестов ставятся в очередь, откуда планировщик их распределяет по серверам. Перечень тестов выдается на экран, после чего команда ожидает окончания тестов и по мере их завершения выдает на экран статус завершения: OK, ERROR или STOP. При этом STOP может означать ошибку компиляции или ошибку симуляции.

Прервать выполнение команды 'kmd qrun' можно нажатием 'CTRL+C', при этом завершится выполнение только скрипта, поставленные в очередь и запущенные задачи будут выполняться дальше.

Полный лог выполнения каждого теста записывается в директорию теста под именем 'qsub_log<суффикс>.txt', где <суффикс> формируется аналогично суффиксу файла 'rtlrun', только вместо точки в качестве разделителя аргументов используется подчеркивание.

Для получения состояния запущенных тестов используется команда 'kmd qstat', выполнять которую следует в отдельном консольном окне. Принудительно завершить тест можно с помощью команды 'kmd qdel'.

Команды 'kmd qrun', 'kmd qstat' и 'kmd qdel' работают с поддиректориями, которые находятся в той папке, из которой запущена команда.

При запуске регрессии на SGE все тесты начинают работать почти одновременно на разных серверах. При этом, если рабочая копия репозитория только что создана или в ней выполнено действие ‘distclean’, компиляция тестов начинается с компиляции и сборки общих библиотек. Очевидно, если несколько десятков тестов одновременно собирают общую библиотеку, конфликтов не избежать. Поэтому, перед запуском регрессии в таком случае необходимо прогнать простой тест, который соберет все библиотеки.

Часто возникает необходимость одновременно запустить на SGE несколько вариантов регрессии одной платформы с разным набором параметров регрессии. Для этого можно применить два подхода. Первый - сделать необходимое количество рабочих копий, настроить их на одну платформу и в каждой выполнить команду ‘kmd qrun <опции>’.

Второй способ - запускать варианты регрессии в одной рабочей копии. Это возможно, т.к. имена директорий ‘rtlrun...’ и имена логфайлов различаются, если заданы разные опции регрессии.

Задача регрессионного тестирования готовых микросхем в составе электронного модуля, точнее блока функционального контроля (БФК), также решается с помощью KMD.

БФК представляет собой одноплатный компьютер, в котором тестируемая микросхема СнК устанавливается в контактирующее устройство, и имеется возможность с расположенной рядом ЭВМ включать и выключать питание, подавать сигнал Reset, загружать в память БФК программу, запускать ее на выполнение и получать всю выдаваемую в консоль информацию. KMD обеспечивает подготовку тестовой программы для БФК.

Используя встроенный скрипт builder можно собрать требуемый набор тестов, готовых к исполнению, и упаковать его в один файл.

Далее этот файл записывается во флешпамять БФК, а таблица расположения тестов в нем передается программе Логгер на инструментальной ЭВМ для автоматического последовательного вызова тестов и анализа результатов их выполнения.

Заключение

Описанный в статье инструмент верификации СнК стал одним из основных при разработке микросхем.

Он задумывался и создавался для конкретной ниши – тестирование нижнего уровня на всех этапах разработки и обеспечение функционального контроля при производстве. На сегодняшний день KMD поддерживает более 20 платформ и включает несколько сотен разнообразных тестов.

Logic Models Test System KMDTESTKIT

S.A.Sidorov, A.B.Slepov

Abstract: Described KMDTESTKIT – Logic model test system for verification of RTL-models of System-on-Chip by software. Illustrated structure and main features of system.

Keywords: testing by software, verification, system-on-chip.

Литература

1. С.Г. Бобков. Методика тестирования микросхем для компьютеров серии «Багет». «Программные продукты и системы, 2007, №3, 2-5.

2. С.Г.Бобков. Маршрут проектирования микросхем серий 1890 и 1990. Часть 1. «Электронные компоненты», 2008, №5, 98-104.

3. С.Г.Бобков. Маршрут проектирования микросхем серий 1890 и 1990. Часть 2. «Электронные компоненты», 2008, №8, 126-131.

4. С.Г. Бобков Высокопроизводительные вычислительные системы. М.: НИИСИ РАН, 2014. – 296 с.

5. Т.С. Титовская, О.В. Непомнящий, А.В. Леонова, А.А. Комаров. Формальная верификация при проектировании сверхбольших интегральных схем. «Вестник КрасГАУ», 2014, №4, 87-89.

6. А.Лохов, А.Рабовалюк. Комплексная функциональная верификация СБИС. Система QUESTA компании Mentor Graphics. «Электроника: Наука, Технология, Бизнес», 2007, №3, 102-109.

7. Л.А. Золотаревич. Верификация проектов и построение тестов контроля СБИС на уровне RTL. «Автоматика и телемеханика», 2013, №1, 146-158.

8. В.Ю. Залетов, Е.А. Янкевич, П.А. Шевченко. Комплексный подход к верификации в процессе разработки отечественной СБИС Декодера цифрового телевизионного сигнала К1879ХБ1Я. https://www.module.ru/upload/images/1354523030complex1.pdf

9. Б. Коллинз-Сассман, Б.У. Фитцпатрик, К.М. Пилато. Управление версиями в Subversion. http://svnbook.red-bean.com/nightly/ru/svn-book.html

10. С.Е.Богомолов. Bog BOS: Sun Grid Engine. http://www.bog.pp.ru/work/sge.html

11. А.С. Куцаев. Генератор стохастических тестов с ограниченным моделированием выполнения. Труды НИИСИ РАН, 2016, том 6, № 1, стр. 69-77.

12. И.В. Грибков, А.В. Захаров, П.П. Кольцов, Н.В. Котович, А.А. Кравченко, А.С. Куцаев, А.С. Осипов, И.Ш. Хисамбеев. Развитие системы стохастического тестирования микропроцессоров Integ. Программные продукты и системы, 2010, №2, с.14-23.

Суперкомпьютер и дискретная топология

Г.Г. Рябов[1], М.Ш.Сургуладзе[2]

[1] *ФГБОУ ВО « Московский государственный университет имени М.В.Ломоносова», Москва, Россия;*

[2] *ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: sfm12@mail.ru*

Аннотация: Для моделирования объектов дискретной топологии привлекаются суперкомпьютерные системы, нередко из списка TOP-500, что вполне объяснимо комбинаторным характером большинства задач, особенно при исследовании динамики перестроек таких структур [2,4]. В статье рассмотрены **три** метода кодирования, которые имеют перспективы развития и связаны с распараллеливанием вычислений в рассматриваемой области: **1.** Кодирование кубических комплексов. **2.** Кодирование для вычисления переходных вероятностей в цепях Маркова при перестройках триангуляций евклидовых пространств. **3.** Кодирование топологических ситуаций при преобразованиях комплексов - дискретных аналогов гомотопных преобразований. База, выбранная для геометрико-топологических объектов, обладает «минимальными» математическими ресурсами (целые точки и примитивные вектора в евклидовых пространствах).

Ключевые слова: дискретная топология, суперкомпьютерные системы, кодирование кубических комплексов, кодирование топологических ситуаций, кодирование для вычисления переходных вероятностей в цепях Маркова.

Введение

Рассматриваются евклидовы пространства $\mathbf{R}^n$ и в них подпространства целых точек $\mathbf{Z}^n$, как вершин объектов (граней, симплексов, комплексов и т.д.) В качестве множества ребер (граней размерности 1) используются отрезки-ребра, инцидентные вершинам в целых точках и коллинеарные примитивным (простым) векторам с целочисленными координатами и не имеющими внутренних целых точек. Если координаты примитивных векторов не превосходят по модулю целого p, то будем обозначать такое множество ребер Vp. Поэтому исходную базу обозначим $\{\mathbf{Z}^n, Vp\}$. В данной статье основные построения будут относиться к случаю $\{\mathbf{Z}^n, V_1\}$.

1. Кодирование кубических комплексов

Вначале рассматриваются все кубические комплексы в рамках n-мерного единичного куба $\mathbf{I}^n$.

Под кубическими комплексами здесь подразумеваются комплексы из k-граней (граней размерности k; $0 \le k \le n$), поскольку каждая грань в $\mathbf{I}^n$ есть «куб» соответствующей размерности.

В [7] на основе анализа свойств коэффициентов пирамиды Паскаля (обобщение треугольника Паскаля для трехмерного случая) установлено свойство **биекции** между множеством всех n-разрядных троичных слов и множеством всех k-граней в $\mathbf{I}^n$.

Пусть в $\mathbf{R}^n$ задан репер $\mathbf{e}_1, \mathbf{e}_2, \ldots \mathbf{e}_n$ и задано $D=d_1,d_2,\ldots d_n$ –n-разрядное троичное слово; d_i из алфавита {0;1;2}. При этом установлено однозначное соответствие между номером разряда в слове и номером реперного вектора $\mathbf{e}_i \rightarrow d_i$.

Выберем некоторую k-грань и положим в D все $d_i=2$, если в выбранной грани содержится единичное ребро, коллинеарное e_i. Так в D будет k «двоек», а остальные разряды будут иметь значения 0 и 1 и определять «место» этой грани в $\mathbf{I}^n$, т.е. задавать параллельный перенос в это «место» (трансляцию).

Определим $\chi_2(D)$ (характеристику размерности) как n-разрядное двоичное слово, разряды q_i которого удовлетворяют соотношению:

$$q_i = \begin{cases} 1, \text{ если } d_i = 2 \\ 0, \text{ в противном случае} \end{cases}.$$

При таком определении $\Sigma\, q_i=k$, где k-размерность грани в $\mathbf{I}^n$.

Определим $\chi_1(D)$ (характеристику трансляции, как n-разрядное двоичное слово, разряды ti которого удовлетворяют соотношению:

$$t_i = \begin{cases} 1, \text{если } d_i = 2 \\ d_i, \text{в противном случае} \end{cases}.$$

Формально можно записать, что грань удовлетворяет соотношению:

$$f(D) = \prod q_i \mathbf{e}_i + \sum t_i \mathbf{e}_i ,$$

где П-перемножение множеств, а $q_i\mathbf{e}_i=\mathbf{e}_i$ при $q_i=1$ и отсутствие $\mathbf{e}_i$ при $q_i=0$. Аналогично для $t_i\mathbf{e}_i$.

Иначе говоря, каждая k-грань «формируется» из набора k реперных ребер

около точки (0,0,…0), а затем транслируется (параллельно сдвигается) в $\mathbf{I}^n$.

Так троичное слово 202211 представляет в $\mathbf{I}^6$ трехмерную грань (3-куб) с ребрами, коллинеарными $\mathbf{e}_1,\mathbf{e}_3,\mathbf{e}_4$ и транслированную на вектор (000011).

Поставим в соответствие комплексу в $\mathbf{I}^n$ из m граней строку из m троичных n-разрядных слов. Так строка 202201;022102; 221200;202101 представляет комплекс в $\mathbf{I}^6$ из трех 3-граней и одной 2-грани. Этот комплекс связный и имеет общую вершину (001101). Это устанавливается поразрядными операциями, определяющими пересечения граней и комплексов и детальное описание которых здесь не рассматривается.

Таким образом формально общее число всех возможных кубических комплексов в $\mathbf{I}^n$ равно мощности множества всех подмножеств граней всех размерностей, т.е. 2^A, где $A=3^n$. Реально их число значительно меньше, поскольку формальная строка может содержать слова, относящиеся к граням, где одна грань содержится в другой. Определение таких поглощений, как и пересечений граней и комплексов также сводится к поразрядным операциям.

В $\mathbf{R}^n$ кубический комплекс в общем случае состоит из кубических комплексов разных $\mathbf{I}^n$, привязанным к координатам (x1,x2,..xn) целых точек. Такая привязка возможна двумя способами.

1-ый способ. Снабдить каждую строку-комплекс из $\mathbf{I}^n$ n-мерной координатой этого $\mathbf{I}^n$ в $\mathbf{R}^n$.

2-ой способ. В n-мерном клеточном массиве в памяти компьютера в каждой клетке поместить строку соответствующего комплекса из $\mathbf{I}^n$.

Во многих случаях, когда перестройка комплексов в каждом $\mathbf{I}^n$ зависит от ситуации в комплексах соседних координат, 2-ой способ имеет конструктивные преимущества.

Такой способ сродни постройке панельного дома с огромным разнообразием топологических «планировок» помещений.

В заключение этого раздела отметим особенность троичного слова (кода грани), которую можно характеризовать как «полуквантовость».

Разряды, которые содержат 2, можно рассматривать как q-бит (из терминологии квантовых вычислений), поскольку за этой двойкой стоят и 1 и 0 и весь отрезок вещественных чисел [0;1] соответствующего ребра из $\mathbf{I}^n$. Разряды со значениями 0 и 1 являются как бы обычными двоичными. Поэтому условно каждое слово-грань=квантовая составляющая (размерность) плюс двоичная составляющая (трансляция).

Дальнейший путь изучения динамики перестроек комплексов (строк троичных слов) связан с введением действий слева симметрической группы Sn или ее подгрупп, на строку как множество троичных слов, и ведет в область алгебраической топологии.

2. Кодирование полного множества элементарных событий и корректное вычисление переходных вероятностей в Марковских цепях.

В [8] рассматриваются примитивные триангуляции $\mathbf{R}^3$ и $\mathbf{R}^4$ также на базе $\{\mathbf{Z}^n,V_1\}$, т.е. триангуляции, которые можно задать различными вариантами расположения диагоналей в гранях куба. Вводятся вектора диагональных степеней вершин, полностью описывющие тип триангуляции куба и инвариантные относительно вращений и отражений. Для каждого такого вектора рассматриваются все возможные варианты перехода его в другой при изменении направления диагонали в одной или нескольких гранях. Для этого строится отображение триангуляции куба в двоичный код для заданного изначально порядка следования граней в развертке куба (плоской для 3d и трехмерной для 4d). «0» в i-ом разряде 6-разрядного(по числу граней) коде для $\mathbf{R}^3$ соответствует диагонали, проведенной в i-ой грани из левого нижнего угла в правый верхний, а «1»-диагонали из правого нижнего угла.

Каждое элементарное событие (перестройка триангуляции) есть отображение одного кода развертки в другой. На всем множестве пар кодов 2^6x2^6 (для $\mathbf{R}^3$) мы определяем корректно, в смысле аксиоматики Колмогорова, переходные вероятности для Марковских цепей, описывающих случайный процесс диагональных перестроек. Эргодические и периодические свойства этих цепей полностью определяются матрицами переходных вероятностей этих процессов. С помощью такого метода кодирования получены стационарные распределения типов триангуляции в $\mathbf{R}^3$ при случайных перестройках одной, двух, трех,…шести диагоналей в кубе за единицу дискретного времени.

Кроме того, исследовалось распределение всех неконгруэнтных вершинных полиэдров при такой триангуляции. Вершинный полиэдр в $\mathbf{R}^3$ образуется из вершинного комплекса при примыкании симплексов из

триангулированных кубов-октантов. Такое распределение отнесенно к кодировке типов полиэдров в структуре Бозе-Эйнштейна. Различные типы (их 5) триангуляции $\mathbf{I}^3$ рассматриваются как энергетические уровни, и частица, соответствующая типу триангуляции каждого из 8 октантов (вокруг вершины) помещается на соответствующий уровень. Таким образом имеем размещение 8 частиц в 5 ящиках (их 495) и для каждого такого «квантового» кода вычислено число неконгруэнтных вершинных полиэдров. Установлено, что в $\mathbf{R}^3$ для пяти таких «квантовых» кодов не существует ни одного типа полиэдра, т.е. спектр разрывен.

3. Кодирование топологических ситуаций при дискретных аналогах гомотопных преобразований

В [1,3,5] предложены методы полиэдризации (разбиения) пространства на полиэдры-зонодроны (одного типа) - дискретный аналог окрестности на базе $\{\mathbf{Z}^n, V_1\}$. На таком построении определяются локальные преобразования расширения и сжатия без склеек и разрывов. Множество таких локальных преобразований и есть дискретный аналог гомотопных преобразований. Самым глубоким циклом при реализации расширения или сжатия является процедура проверки отсутствия разрыва или склейки, т.е. корректности каждого локального топологического действия. Такая процедура сводится к определению связных компонент на вершинном полиэдре. При однородной полиэдризации $\mathbf{R}^3$ с помощью кубододекаэдра с числом вершин 14 и числом ребер 50 во-первых надо хранить лишь один эталон полиэдра (ребра виртуальны), а во -вторых анализ топологической корректности (примерно эквивалентен 700 операций) заменяется табличным хранением исходов всех вариантов, которые заранее рассчитаны т.е. 2^{14} битов (затраты по памяти -мегабайт).

Это дает сокращение машинного времени примерно в 1000 раз для $\mathbf{R}^3$, а для $\mathbf{R}^4$ при числе вершин 30 и числе ребер более 10000 раз (расходы по памяти- Гбайт).

В определенном смысле идет размен скорости на емкость памяти. При такой организации процесс гомотопного расширения множества до объема решетки 10^9(1000x1000x1000) вершин и числа ребер 10^{13} на суперкомпьютере МГУ «Чебышев» (60 терафлопс) идет несколько минут, на суперкомпьютере МСЦ-МВК-100К (5 терафлопс) - 95 минут.

Заключение

Приведенные примеры взяты как фрагменты методов, используемых при создании инструментального комплекса [6] для операций с объектами дискретной топологии (общее название «топологический процессор»), ориентированного на современные суперкомпьютеры массивно-параллельной архитектуры. Примеры показывают дополнительные возможности эффективного использования суперкомпьютеров при подходе, который можно было бы характеризовать по аналогии с геометрией чисел, как «топология кодов».

1. Такое кодирование допускает во многих случаях поразрядные операции, что близко к предельным схемам распараллеливания вычислений до разряда, что особенно важно при высоких размерностях пространств.
2. Предложенные методы могут найти свое аппаратное воплощение в виде сопроцессоров к суперкомпьютеру.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП14) по теме (проекту) «36.18. Исследование и разработка моделей и методов математического и компьютерного моделирования тепловых процессов в электронных системах различного назначения в условиях неопределенности». (0065-2014-0018).

.

Supercomputers and discrete topology

G.G. Ryabov, M.Sh. Surguladze

Abstract: For simulation of objects of discrete topology, supercomputers are often used. This can be explained by the combinatorial nature of the problems examined, especially in the study of the dynamics of rearrangements of such structures. In this paper, we consider three methods of coding associated with the parallelization of computations: (1) coding of cubic complexes; (2) coding for the calculation of transition probabilities in Markov chains in the reconstructions of triangulations of Euclidean spaces; (3) coding of topological situations in the transformations of complexes, i.e., discrete analogs of homotopic transformation. The base chosen for the geometric and topological objects possesses “minimal” mathematical resources (integer points and primitive vectors in Euclidean spaces).

Keywords: discrete topology, Supercomputer systems, Encoding of cubic complexes, Encoding topological situation, Encoding for calculation of transition probabilities in Markov chains

Литература

1. M. Couprie, G. Bertrand Simplicity surfaces: a new definition of surfaces in Z3.«SPIE Proceedings», vol.3454, 1998

2. M. Latapy. Generalized integer partitions, tilings of zonotopes and lattices. arXiv:math/0008022v2[math.CO], 2000.

3. Y. Kenmochi, A. Imiya. Discrete polyhedrization of lattice point set.«LNCS Proceedings» Vol. 2243, 2001.

4. V. Desoutter, N. Destainville. Flip dynamics in three- dimensional random tilings. arXiv:cond-mat/0406728v3[cond-mat.stat-mech] 2004.

5. Г.Г. Рябов. Метрические и топологические волны на решетках. Изд. МГУ, 2005.

6. G. Ryabov, V. Serov, Simplicial-lattice model and metric-topological constructions. «PRIP Proceedings. – Minsk, 22-24 May 2007», vol II.

7. Г.Г. Рябов. О путевом кодировании k-граней в n-кубе.«Вычислительные методы и программирование». т. 9 (2008), № 1, 16 – 18.

8. Г.Г. Рябов. Цепи Маркова в динамике примитивной триангуляции R3 и R4.«Вычислительные методы и программирование». т.10 (2009), № 1, 1 – 8.

Ориентация объектов в системах виртуального окружения

М.В. Михайлюк[1], Е.В. Страшнов[2], Д.А. Кононов[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1]mix@niisi.ras.ru, [2]strashnov_evg@mail.ru, [3]dmitrykon52@gmail.com

Аннотация: Одной из важных составляющих в комплексах управления объектами в системах виртуального окружения является задание ориентации объектов. Для задания ориентации летательных аппаратов, кораблей, автомобилей и других средств передвижения часто используются углы Эйлера, т.к. расположение движителей на этих средствах позволяет поворачивать их вокруг своих осей координат. Однако при этом возникает так называемая проблема блокировки кардана. Когда один угол поворота равен 90^o, то невозможно однозначно определить остальные углы Эйлера для такой ориентации. Для решения этой проблемы в работе предлагается использовать дополнительный (четвертый) угол. Показывается, что такое использование позволяет однозначно задать ориентацию и избежать блокировки кардана.

Ключевые слова: ориентация, матрица поворота, углы Эйлера, блокировка кардана

Введение

В комплексах управления сложными динамическими объектами в системах виртуального окружения [1-3] важной составляющей является задание ориентации объектов. Для этой цели часто принято использовать углы Эйлера. Особенно удобны эти углы для задания ориентации средств перемещения (летательных аппаратов, судов, автомобилей и т.д.), так как расположение движителей на этих средствах позволяет поворачивать их вокруг фиксированных осей их локальных систем координат. Однако при использовании углов Эйлера может возникнуть так называемая проблема блокировки кардана (gimbal lock). Если один из углов поворота равен 90^o, то невозможно однозначно восстановить остальные углы, т.е. однозначно задать ориентацию объекта.

Для решения этой проблемы исследователи предлагают либо ограничить области изменения углов, либо принудительно задавать одну из возможных ориентаций объекта [4]. Такие подходы не обеспечивают универсальность системы управления. Для однозначного задания ориентации можно использовать кватернионы. Однако переход от кватерниона к углам Эйлера также содержит проблему неоднозначности. Здесь предлагается ввести дополнительный (четвертый) угол поворота. В работе показывается, что в этом случае любая ориентация задается однозначно и проблема блокировки кардана не возникает.

1. Задание ориентации объекта

Пусть в трехмерном векторном пространстве задан объект и связанная с ним система координат $OXYZ$. Осуществим произвольный поворот объекта вокруг точки O. Тогда его система координат перейдет в систему $OX'Y'Z'$. Каждое положение системы координат объекта можно рассматривать как его ориентацию. Таким образом, системы координат $OXYZ$ и $OX'Y'Z'$ задают начальную и конечную ориентацию объекта. Рассмотрим произвольный вектор $\vec{v} = \left(x_v, y_v, z_v\right)^T$ в начальной системе координат и обозначим через $\vec{v}' = \left(x_v', y_v', z_v'\right)^T$ координаты этого вектора в той же систем после поворота. Как известно [5], соотношение между этими координатами можно задать с помощью некоторой матрицы M размера 3x3 так, что

$$\vec{v}' = M \cdot \vec{v}. \qquad (1)$$

Матрица M называется матрицей поворота. Столбцами матрицы M будут координаты векторов X', Y' и Z' в системе $OXYZ$ (см. [6]). Матрица M будет задавать ориентацию

объекта относительно его начальной ориентации.

Кроме матрицы ориентацию объекта можно задать также с помощью углов Эйлера. Именно, повернув объект сначала на некоторый угол ψ вокруг оси Y, затем на некоторый угол θ вокруг оси Z (ее нового положения после первого поворота) и наконец, на некоторый угол φ вокруг оси X (ее нового положения после двух поворотов). Области изменения этих углов следующие: $\varphi,\psi \in [0,2\pi)$, $\theta \in [-\pi/2,\pi/2]$. Оси можно выбрать в произвольном порядке, главное, чтобы они были различны или одинаковыми могут быть только оси первого и третьего поворотов (см. [7]).

Рассмотрим переход от одного способа задания ориентации к другому. Пусть заданы углы Эйлера φ,θ,ψ. Матрицы поворота вокруг соответствующих осей на эти углы имеют вид:

$$R_\varphi = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_\varphi & -s_\varphi \\ 0 & s_\varphi & c_\varphi \end{pmatrix},$$

$$R_\theta = \begin{pmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$R_\psi = \begin{pmatrix} c_\psi & 0 & s_\psi \\ 0 & 1 & 0 \\ -s_\psi & 0 & c_\psi \end{pmatrix}.$$

Здесь символами c_φ и s_φ обозначаются косинусы и синусы угла φ (аналогично для остальных углов). Результирующая матрица поворота будет иметь вид:

$$R = R_\psi R_\theta R_\varphi =$$

$$= \begin{pmatrix} c_\psi c_\theta & -c_\psi c_\varphi s_\theta + s_\varphi s_\psi & c_\psi s_\theta s_\varphi + s_\psi c_\varphi \\ s_\theta & c_\varphi c_\theta & -s_\varphi c_\theta \\ -s_\psi c_\theta & s_\psi c_\varphi s_\theta + s_\varphi c_\psi & -s_\psi s_\theta s_\varphi + c_\psi c_\varphi \end{pmatrix}.$$ Это и будет искомая матрица M.

Пусть, обратно, задана матрица $M = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$. Попробуем найти соответствующие ей углы Эйлера φ,θ,ψ. Так как $M = R$, то $m_{2,3} = -s_\varphi c_\theta$, Если $c_\theta = 0$, то $s_\theta = \pm 1$ и матрица M приобретает следующий вид:

$$M = \begin{pmatrix} 0 & -c_\psi c_\varphi s_\theta + s_\varphi s_\psi & c_\psi s_\theta s_\varphi + s_\psi c_\varphi \\ s_\theta & 0 & 0 \\ 0 & s_\psi c_\varphi s_\theta + s_\varphi c_\psi & -s_\psi s_\theta s_\varphi + c_\psi c_\varphi \end{pmatrix}.$$

При $s_\theta = 1$ получаем

$$M = \begin{pmatrix} 0 & -c_\psi c_\varphi + s_\varphi s_\psi & c_\psi s_\varphi + s_\psi c_\varphi \\ 1 & 0 & 0 \\ 0 & s_\psi c_\varphi + s_\varphi c_\psi & -s_\psi s_\varphi + c_\psi c_\varphi \end{pmatrix} =$$

$$= \begin{pmatrix} 0 & -c_{\varphi+\psi} & s_{\varphi+\psi} \\ 1 & 0 & 0 \\ 0 & s_{\varphi+\psi} & c_{\varphi+\psi} \end{pmatrix}.$$

Мы видим, что углы φ,ψ невозможно однозначно определить из этой матрицы. Такая же ситуация будет в случае, когда $s_\theta = -1$. Эта проблема называется блокировкой кардана (см. [8]). Она возникает при любой последовательности углов Эйлера.

Таким образом, если ориентация задана в виде матрицы, то восстановить ее однозначно в виде последовательности углов Эйлера не представляется возможным, если соответствующий угол оказывается прямым.

2. Дополнительный угол поворота

Для решения проблемы блокировки кардана предлагается ввести дополнительный **контролируемый** угол χ поворота вокруг оси Z, величину которого можно менять так, чтобы $c_\theta \neq 0$.

Более точно, мы вводим последовательность поворотов $R_\chi R_\psi R_\theta R_\varphi$ вокруг осей соответственно Z,Y,Z,X. Угол χ равен нулю до тех пор, пока $\theta \in [-85^o, 85^o]$.

В этом случае все углы Эйлера однозначно вычисляются по матрице M.

Если же угол θ выходит за эти пределы (т.е. c_θ приближается к нулю), мы увеличиваем угол χ.

Это вызывает изменение всех остальных углов, и угол θ опять входит в пределы $[-85^o, 85^o]$.

Это позволяет однозначно определить все углы. Опишем это более подробно.

Для четырех углов поворота матрица $R = M$ будет иметь следующий вид:

$$R = R_\chi R_\psi R_\theta R_\varphi = \begin{pmatrix} c_\chi c_\psi c_\theta - s_\chi s_\theta & c_\chi(-c_\psi s_\theta c_\varphi + s_\psi s_\varphi) - s_\chi c_\theta c_\varphi & c_\chi(c_\psi s_\theta s_\varphi + s_\psi c_\varphi) + s_\chi c_\theta s_\varphi \\ s_\chi c_\psi c_\theta + c_\chi s_\theta & s_\chi(-c_\psi s_\theta c_\varphi + s_\psi s_\varphi) + c_\chi c_\theta c_\varphi & s_\chi(c_\psi s_\theta s_\varphi + s_\psi c_\varphi) - c_\chi c_\theta s_\varphi \\ -s_\psi c_\theta & s_\psi s_\theta c_\varphi + c_\psi s_\varphi & -s_\psi s_\theta s_\varphi + c_\psi c_\varphi \end{pmatrix}.$$

Нам надо показать, что всегда существует угол χ такой, что φ,θ,ψ однозначно определяются из матрицы $R = M$. Из элементов матрицы можно составить систему линейных уравнений

$$\begin{cases} m_{11} = c_\chi c_\psi c_\theta - s_\chi s_\theta \\ m_{21} = s_\chi c_\psi c_\theta + c_\chi s_\theta \end{cases}$$

относительно неизвестных $c_\psi c_\theta$ и s_θ (напомним, что коэффициенты c_χ и s_χ этой системы нам известны). Детерминант системы равен

$$D = \left(c_\chi\right)^2 + \left(s_\chi\right)^2 = 1.$$

Так как он для любого χ отличен от нуля, то система для любого χ будет иметь единственное решение для $c_\psi c_\theta$ и s_θ. Всегда можно взять такое значение χ, чтобы $\theta \in [-85^o, 85^o]$. Тогда $s_\theta \in (-1,1)$, $c_\theta \in (0,1]$, поэтому $c_\theta = +\sqrt{1-\left(s_\theta\right)^2}$ и $\theta = \mathrm{a}rctg(s_\theta / c_\theta)$.

Далее, $m_{31} = -s_\psi c_\theta$, откуда $s_\psi = -m_{31}/c_\theta$. Разделив значение $c_\psi c_\theta$ на c_θ, получаем значение c_ψ.

Для вычисления синуса и косинуса угла φ рассмотрим еще одну систему линейных уравнений, которая определяется элементами m_{12} и m_{22} матрицы M

$$\begin{cases} m_{12} = c_\chi(-c_\psi s_\theta c_\varphi + s_\psi s_\varphi) - s_\chi c_\theta c_\varphi \\ m_{22} = s_\chi(-c_\psi s_\theta c_\varphi + s_\psi s_\varphi) + c_\chi c_\theta c_\varphi \end{cases}$$

относительно неизвестных s_φ и c_φ. Ее можно переписать в виде

$$\begin{cases} m_{12} = c_\chi s_\psi s_\varphi - (s_\chi c_\theta + c_\chi c_\psi s_\theta) c_\varphi \\ m_{22} = s_\chi s_\psi s_\varphi + (c_\chi c_\theta - s_\chi c_\psi s_\theta) c_\varphi \end{cases}.$$

Детерминант этой системы равен

$$D_1 = c_\chi s_\psi (c_\chi c_\theta - s_\chi c_\psi s_\theta) +$$
$$+ s_\chi s_\psi (s_\chi c_\theta + c_\chi c_\psi s_\theta) =$$
$$= s_\psi c_\theta (c_\chi^2 + s_\chi^2) = s_\psi c_\theta.$$

Так как $c_\theta > 0$, то, в случае $s_\psi \neq 0$, система имеет единственное решение для s_φ и c_φ. Если же $s_\psi = 0$, то $m_{32} = c_\psi s_\varphi$ и $m_{33} = c_\psi c_\varphi$, откуда получаем

$$s_\varphi = m_{32}/c_\psi \quad \text{и} \quad c_\varphi = m_{33}/c_\psi.$$

Таким образом, любую ориентацию можно однозначно задать с помощью четырех углов поворота.

Заключение

В результате данного исследования предложен метод обхода проблемы блокировки кардана при построении углов Эйлера по матрице ориентации путем добавления дополнительного угла поворота. Это позволяет улучшить управление объектами в системах виртуального окружения и имитационно-тренажерных комплексах. Апробация предложенного метода в рамках комплекса управления системы VirSim [9-10], разработанной в ФГУ ФНЦ НИИСИ РАН, показала его адекватность поставленным задачам.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) «34.9. Системы виртуального окружения: технологии, методы и алгоритмы математического моделирования». (0065-2016-0005).

Objects orientation in virtual environment systems

M.V. Mikhaylyuk, E.V. Strashnov, D.A. Kononov

Abstract: Setting the orientation of objects is one of the important components of object control complexes in virtual environment systems. Euler angles are often used to specify the orientation of aircrafts, ships, cars and other vehicles, as the location of the propulsions allows these vehicles to be rotated about their coordinate axes. However, the so-called gimbal lock problem arises. When one rotation angle is 90 degrees, other Euler angles for this orientation are impossible to be definitely determined. To solve this problem, an additional (forth) angle is proposed to use. It is shown that such usage allows the orientation to be definitely specified and avoids the gimbal lock.

Keywords: orientation, rotation matrix, Euler angles, gimbal lock.

Литература

1. А.В. Мальцев, П.Ю. Тимохин. Эргономичный интерфейс управления виртуальной средой с использованием распределенных вычислений на GPU // Вестник кибернетики, 2017, № 2, с. 69-74.
2. А.В. Мальцев, М.В. Михайлюк. Методы эргономичного управления объектами и параметрами виртуальной среды // Труды НИИСИ РАН, 2017, том 7, № 1, с. 41-45.
3. М.В. Михайлюк, Д.В. Омельченко, Е.В. Страшнов. Командный и супервизорный режимы управления виртуальными роботами // Вестник кибернетики, 2016, № 4, с. 79-84.
4. G.G. Slabaugh. Computing Euler angles from a rotation matrix, City Univ. London, London, U.K., Technical Report, 1999.
5. A.L. Schwab, J.P. Meijaard. How to draw Euler angles and utilize Euler parameters, In Proc. IDEETC/CIE 2006, ASME 2006 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, 2006.
6. Курс теоретической механики: Учебник для вузов/ В.И. Дронг, В.В. Дубинин, М.М. Ильин и др.; Под общ. Ред. К.С.Колесникова, 3-е изд., стереотип.,
М.: Изд-во МГТУ им. Н.Э. Баумана, 2005, 736 с.
7. J. Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors, Technical Report, Stanford University, 2006.
8. F.S. Grassia. Practical Parameterization of Rotations using Exponential Map, Journal of Graphics Tools, 3(3), pp. 29-48, 1998.
9. М.В. Михайлюк, М.А. Торгашев. Система визуализации "GLView" для имитационно-тренажерных комплексов подготовки космонавтов // Пилотируемые полеты в космос, № 4, 2013, с. 60-74.
10. М.В. Михайлюк, В.И. Брагин. Технологии виртуальной реальности в имитационно-тренажерных комплексах подготовки космонавтов // Пилотируемые полеты в космос, ФГБУ «НИИ ЦПК имени Ю.А.Гагарина», № 2(7), 2013 стр. 82-93.

Использование z-буфера для поиска столкновений частиц с объектами 3D сцены

А.В. Мальцев[1], Е.В. Страшнов[2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1]avmaltcev@mail.ru , [2]strashnov_evg@mail.ru,

Аннотация: В работе предлагается оригинальный подход к решению задачи поиска коллизий систем мелкоразмерных частиц с объектами трехмерных виртуальных сцен с помощью z-буфера (буфера глубины) видеоадаптера. В основе рассматриваемых решений лежит применение распределенных вычислений на современных CUDA-совместимых графических процессорах. Анализ наличия коллизий частиц с объектами, а также вычисление координат точек столкновений в мировом пространстве выполняется в масштабе реального времени.

Ключевые слова: система частиц, трехмерная сцена, столкновение, виртуальная камера, распределенные вычисления, графический процессор.

Введение

В процессе компьютерного моделирования и визуализации трехмерных сцен, как правило, возникает задача реализации взаимодействия между собой виртуальных объектов. Наиболее часто встречающимся видом взаимодействий объектов являются их столкновения, которые в дальнейшем мы будем также называть коллизиями. Решение упомянутой задачи состоит из нескольких основных этапов: определения факта наличия коллизии, вычисления точек соприкосновения объектов и обработки коллизии. Последняя заключается в моделировании поведения объектов после произошедшего столкновения.

Существенную роль в трудности выполнения данных этапов, а также выборе применяемых при этом вычислительных методов и алгоритмов играет сложность геометрии моделей в виртуальной сцене. Чтобы ускорить процесс определения коллизий и поиск точек столкновения, трехмерные объекты принято заменять описывающими их аппроксимирующими контейнерами. В качестве таких контейнеров обычно выступают геометрические примитивы: сферы, параллелепипеды, цилиндры и т.д. [1, 2].

Особым классом являются объекты и некоторые природные явления без четких геометрических границ, сформированные множеством мелкоразмерных элементов (снег, дождь, струи жидкости, пылевые облака и т.п.). Как правило, подобные объекты моделируются в виртуальной среде системами частиц [3-5]. Число элементов в таких системах может достигать сотен тысяч, а иногда и нескольких миллионов. При этом задача определения столкновений системы частиц с объектами трехмерной сцены является весьма трудоемкой. Использование упомянутого подхода с аппроксимирующими контейнерами в данном случае затруднительно и не позволяет обеспечить ее решение в масштабе реального времени, необходимом для правильной работы имитационно-тренажерных комплексов и систем виртуального окружения.

В данной статье предлагаются новые методы и алгоритмы распределенного поиска коллизий систем мелкоразмерных частиц (СМЧ) с объектами трехмерных виртуальных сцен в масштабе реального времени. Задача решается для таких СМЧ, частицы которых движутся прямолинейно по траекториям, близким к параллельным (например, капли ливневого дождя или снежные хлопья при снегопаде в безветренную погоду). Предлагаемые подходы основаны на применении многоядерных графических процессоров (GPU) с поддержкой архитектуры параллельных вычислений CUDA, а также использовании принципов работы z-буфера видеоадаптера.

1. Поиск коллизий на GPU

Пусть эмиттер E рассматриваемой СМЧ имеет прямоугольную форму и размер W x H, а испускаемые из него частицы движутся в направлении $\boldsymbol{D} \perp E$. Для осуществления поиска столкновений частиц с объектами трехмерной сцены разместим в эмиттере E жестко связанную с ним виртуальную камеру C. Начало ее системы координат (СК) VCS расположим в точке P_e – центре прямоугольника E, ось Y (задает верх камеры)

направим вдоль меньшей стороны E, ось Z – противоположно вектору $\boldsymbol{D}$. Направление взгляда камеры в таком случае будет совпадать с $\boldsymbol{D}$. Используя функцию *glOrtho* из графической библиотеки OpenGL, установим для C ортографическое проецирование с такими параметрами, чтобы область видимости камеры охватывала область распространения частиц. Для этого зададим соответственно левую, правую, нижнюю и верхнюю плоскости отсечения по границам эмиттера E, а именно: $l_p = -W/2$, $r_p = W/2$, $b_p = -H/2$, $t_p = H/2$. Ближняя плоскость отсечения при этом должна находится на небольшом расстоянии n_p от эмиттера (например, $n_p = 0.1$), а дальняя – на расстоянии f_p, не меньшем длины максимального пути произвольной частицы СМЧ за время ее жизни.

Во время синтеза каждого кадра изображения виртуальной сцены, перед расчетом состояния СМЧ, будем визуализировать сцену из камеры C с включенным тестом глубины, но блокированной записью в буфер цвета (блокировка производится с помощью функции *glColorMask*). В результате такого рендера z-буфер содержит карту глубины сцены относительно эмиттера СМЧ. Напомним, что для камеры C было установлено ортографическое проецирование с боковыми плоскостями отсечения по границам эмиттера СМЧ. Поэтому после выполнения данного рендера каждый пиксел z-буфера будет хранить расстояние от соответствующей ему точки эмиттера E до ближайшего к E объекта по лучу, параллельному взгляду камеры и направлению $\boldsymbol{D}$ движения частиц. В том случае, если луч не пересекается ни с одним объектом до дальней плоскости отсечения, то значение пиксела соответствует расстоянию до этой плоскости.

Поскольку в дальнейшем информация из z-буфера нам потребуется в виде текстуры, то описанный рендеринг необходимо выполнять не в стандартный буфер глубины текущего контекста, а сразу в текстуру T_{depth} с форматом *GL_DEPTH_COMPONENT32F* (соответствует формату данных z-буфера). Прямую визуализацию в текстуру можно осуществить с использованием технологии FBO (*framebuffer object*, [6]).

Подготовленную карту глубины T_{depth} необходимо конвертировать в текстуру T_r CUDA-совместимого формата (см. п.2) и передать в CUDA-программу (ядро) расчета параметров СМЧ (методы и алгоритмы такого расчета на графическом процессоре в реальном времени подробно изложены в работе [7]). Каждая частица при этом обрабатывается отдельным потоком на GPU. В этом же потоке будем искать возможные коллизии частицы с объектами виртуальной сцены, а также моделировать поведение частицы после столкновения.

Рассчитав в текущий момент времени новое положение P_{wcs} и скорость $\boldsymbol{v}$ некоторой

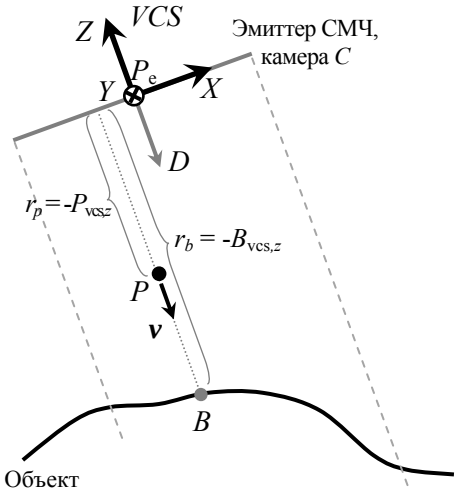

Рис. 1. Сравнение глубины частицы с глубиной, записанной в z-буфере

частицы P в мировой системе координат WCS, поток определяет ее координаты P_{vcs} (рис. 1) в видовой СК VCS виртуальной камеры C. Расстояние r_p от эмиттера E до частицы, равное отрицательному значению z-координаты точки P_{vcs}, сравнивается с расстоянием r_b от эмиттера до точки B некоторого объекта сцены по лучу, проходящему через P и перпендикулярному E. Значение r_b считывается потоком из переданной ему ранее текстуры T_r. Если $r_b \le r_p$, то частица расположена глубже относительно эмиттера, чем объект сцены, следовательно, имеет место коллизия частицы и объекта. В таком случае, необходимо определить координаты точки столкновения и выполнить обработку коллизии. Алгоритм работы вычислительного потока более подробно будет описан в п.3.

2. Конвертация карты глубины

Доступ к текстурам OpenGL в параллельных потоках CUDA осуществляется с помощью механизма *CUDA OpenGL interoperability*, разработанного компанией NVIDIA. Однако пока он не поддерживает текстуры с форматом данных *GL_DEPTH_COMPONENT32F*. Поэтому, чтобы передать информацию из z-буфера, сохраненную в T_{depth}, необходимо конвертировать эту текстуру в один из поддерживаемых в CUDA форматов. Для решения нашей задачи больше всего подходит формат *GL_R32F* – текстура с одним каналом, в которой каждый пиксел хранит 32-битное вещественное число.

Саму конвертацию предлагается выполнять с помощью фрагментного шейдера, написанного на языке GLSL. Данный шейдер принимает на вход текстуру T_{depth} и осуществляет рендеринг с использованием FBO в новую текстуру T_r идентичного размера (в пикселах), но с форматом *GL_R32F*.

Кроме изменения формата данных, выполним в упомянутом шейдере еще одно преобразование. Некоторое число d, записанное в произвольном пикселе T_{depth}, представляет собой расстояние от эмиттера E до точки B ближайшего к нему объекта виртуальной сцены по лучу, перпендикулярному E и проходящему через данный пиксел.

Значение d измерено в единицах экранной СК, связанной с камерой C, и лежит в отрезке [0.0, 1.0].

В рассматриваемом подходе необходимо знать расстояние r_b от эмиттера СМЧ до точки B по оси Z видовой СК VCS камеры C (рис. 1). Поскольку визуализация карты глубины выполнялось с использованием ортографического проецирования, то

$$r_b = -B_{VCS,Z} = n_P + d\left(f_P - n_P\right),$$

где B_{vcs} – точка B с координатами, выраженными в СК VCS, n_p и f_p – расстояния соответственно до ближней и дальней плоскостей отсечения виртуальной камеры C. Расстояния r_b вычисляются на графическом процессоре параллельно для всех пикселов текстуры T_{depth} и записываются в соответствующие пикселы текстуры T_r.

3. Алгоритм работы CUDA-потока

В процессе вычисления параметров СМЧ каждый поток на GPU отвечает за свою частицу. При этом рассчитываются ее параметры (положение, скорость, ускорение, размер и т.д.) в текущий момент времени. После определения положения P_{wcs} некоторой частицы P в мировой системе координат WCS, поток выполняет поиск возможной коллизии этой частицы с каким-либо из объектов виртуальной сцены. Для этого вычисляется расстояние r_p от эмиттера E до частицы (рис. 1):

$$r_p = -P_{\text{vcs,z}}, \quad P_{\text{vcs}} = M_{\text{v}} \cdot P_{\text{wcs}},$$

где M_{v} – видовая матрица камеры, переводящая из СК WCS в СК VCS. Также производится расчет экранных координат P_{s} частицы по формуле:

$$P_{\text{s}} = M_{\text{b}} \cdot M_{\text{pr}} \cdot P_{\text{vcs}},$$

где M_{pr} – матрица ортографического проецирования, заданного в п.1, которая выполняет преобразование из СК VCS в СК нормализованного объема видимости (NDCS). Матрица

$$M_{\text{b}} = \begin{pmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

осуществляет сдвиг начала СК NDCS в вершину (-1, -1, -1) куба видимости и растягивает ее оси в 2 раза.

Далее потоком производится чтение из текстуры T_r расстояния r_b от эмиттера до ближайшего объекта виртуальной сцены по линии движения частицы P. Обращение к T_r выполняется по паре координат ($P_{s,x}$, $P_{s,y}$). Полученное значение r_b сравнивается с рассчитанным ранее расстоянием r_p. При $r_b \leq r_p$ поток фиксирует наличие столкновения частицы с объектом виртуальной сцены. Координаты точки B столкновения в мировой СК WCS при этом определяются по формуле:

$$B_{\text{wcs}} = M_{\text{v}}^{-1}\left(P_{\text{vcs},x}, P_{\text{vcs},y}, -r_b\right).$$

Методы и алгоритмы обработки коллизий зависят от задачи, решаемой при моделировании какого-либо объекта или природного явления с помощью системы частиц, а также от типа взаимодействующих объектов. Например, частицы падающего снега, столкнувшиеся с чем-либо, можно просто удалять из СМЧ, а для капель дождя моделировать разбрызгивание при ударе о поверхность.

Заключение

Описанные в статье методы и алгоритмы основаны на применении распределенных вычислений с помощью современных многоядерных CUDA-совместимых графических процессоров. Это позволяет обеспечить поиск коллизий СМЧ с другими объектами трехмерных виртуальных сцен в масштабе реального времени. Предложенные решения были реализованы в виде программных модулей, которые прошли апробацию в составе разработанной в ФГУ ФНЦ НИИСИ РАН системы визуализации «GLView» [8].

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 16-07-00796.

Using z-buffer to find collisions of particles with 3D scene objects

A.V. Maltsev, E.V. Strashnov

Abstract: At this paper the novel approach is proposed to solve a collision detection task of particle systems with objects of three-dimensional virtual scenes by means of video card's z-buffer (depth buffer). The base of solutions under consideration is application of distributed computing on modern CUDA-compatible graphics processors. Existence analysis of particle collisions with objects and also calculation of collision points' coordinates in world space are performed at real time.

Keywords: particle system, three-dimensional scene, collision, virtual camera, distributed calculations, graphics processor.

Литература

1. А.М.Трушин, М.В.Михайлюк. Определение коллизий аппроксимирующих капсул и сфер трехмерных виртуальных динамических объектов // Труды НИИСИ РАН, 2016, Т. 6, № 2, С. 42-45.
2. Ming C. Lin, S.Gottschalk S. Collision detection between geometric models: a survey // Proc. of IMA conference on mathematics of surfaces, 1998, vol. 1, P. 602-608.
3. J.Tan, X.Fan. Particle system based snow simulating in real time // Procedia Environmental Sciences 10, 2011, vol. 10, p. 1244-1249.
4. W.T.Reeves. Particle systems – a technique for modeling a class of fuzzy objects // In Computer Graphics (Proc. SIGGRAPH), 1983, p. 359-376.
5. А.В.Мальцев. Методы моделирования на GPU снега и дождя в имитационно-тренажерных комплексах // Труды НИИСИ РАН, 2015, т. 5, № 2, с. 42-46.
6. Framebuffer Object // URL: https://www.khronos.org/opengl/wiki/Framebuffer_Object (дата обращения: 14.02.2018)
7. А.В.Мальцев. Реализация системы частиц в реальном времени на GPU // Программные продукты и системы, 2014, № 4, с. 57-62.
8. М.В.Михайлюк, М.А.Торгашев. Система визуализации «GLView» для имитационно-тренажерных комплексов и систем виртуального окружения // Труды 25-й Международной научной конференции Графикон, 2015, с. 96-101.

Необходимость учёта изменения смачиваемости пород при моделировании тепловых методов добычи нефти

В.А.Юдин [1], И.В.Афанаскин [2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1] yudinval@yandex.ru, [2] ivan@afanaskin.ru

Аннотация. Рассмотрена возможность изменений смачиваемости пород при применении тепловых методов добычи нефти. Приведён краткий обзор результатов экспериментальных исследований зависимости характера смачиваемости от температуры. Карбонатные породы с повышением температуры становятся более гидрофильными, результаты по песчано-глинистым породам - крайне противоречивы. На простых модельных расчётах проанализировано влияние изменения смачиваемости на динамику разработки, в сравнении с изменением вязкости нефти. Показано, что изменение смачиваемости может как значительно улучшать показатели разработки (в гидрофобных породах), так и заметно ухудшать их (в гидрофильных). Сделан вывод о необходимости детальных исследований в этом направлении, и тщательном рассмотрении возможных изменений смачиваемости при моделировании тепловых методов добычи нефти. Без этого численное моделирование добычи может быть весьма недостоверным.

Ключевые слова: тепловые методы добычи нефти, методы увеличения нефтеотдачи, смачиваемость, моделирование разработки.

Введение

По данным большинства аналитиков, производство и потребление энергии в мире в ближайшие 2–3 десятилетия будет расти, причём вклад нетрадиционных и возобновляемых источников энергии в общее производство и потребление энергии через 20–30 лет составит только около 30% [3, 4].

Вклад не возобновляемых углеводородных источников энергии (нефти, газа, угля) будет определяющим в снабжении человечества энергией, в том числе и России [1–3].

На территории России, составляющей 12,8% территории Земли, сосредоточено 12-13% прогнозных ресурсов и около 12% разведанных запасов нефти [1–4].

На сегодняшний день суммарные ресурсы нефти и битумов оцениваются в 2,066 млрд.т [1].

Однако, доказанные запасы нефти в России не велики, и при сохранении нынешних темпов её добычи их хватит менее чем на 30 лет [3].

Ввиду неблагоприятного соотношения между разведанными запасами и уровнем годовой добычи, многие специалисты прогнозируют возможное значительное падение нефтедобычи в России в недалёком будущем [1-3].

Помимо этого, наблюдается и ухудшение качества остаточных доказанных запасов [1, 2, 4].

1. Вязкие нефти как один из потенциальных источников нефтедобычи в России

По данным И.С. Гольдберга (1981), суммарные прогнозные ресурсы вязких нефтей, нефтяных песков и битумов на территории бывшего СССР составляют 30-33 млрд.т. [5], из них ресурсы тяжёлых нефтей составляют 13,4 млрд.т. [6]. По данным работы [7], от общего объёма разведанных в России запасов нефти 14% - приходится на тяжёлые нефти, 11% - на высоковязкие. Они сосредоточены в трёх основных провинциях - Волго-Уральской, Западно-Сибирской и Тимано-Печорской [7], причём 60% запасов тяжёлых нефтей сосредоточено в 15 месторождениях: Русское, Ван-Еганское, Фёдоровское, Ново-Елховское, Усинское, Торавейское, Северо-Комсомольское, Ново-портовское, Комсомольское, Вынгапуровское, Западно-Мессояхское, ТазовскоеНаульское, Ярегское, Медынское-Море, Приразломное, Сюрхаратинское и др. [7, 8].

Основная трудность в разработке месторождений вязких нефтей - низкая скорость фильтрации, обусловленная их высокой вязкостью. Применение традиционных методов – например заводнения - не позволяет достичь высокого значения коэффициента извлечения нефти ввиду вязкостной неустойчивости фронта вытеснения

вязкой нефти водными или газовыми вытесняющими агентами.

При этом практически для всех таких нефтей вязкость существенно снижается с ростом температуры, обычно по одному из известных законов [10, 11]:

$$ln(\mu) = A - B \cdot ln(T),\ B > 0,$$
$$ln(\mu) = A - B/(T + C),\ B > 0$$
$$ln(\mu + A) = B \cdot T^{-C},$$

где μ - вязкость нефти, T - температура, A, B и C - экспериментальные коэффициенты.

Поэтому основными методами разработки таких месторождений являются тепловые. Они основаны на различных способах повышения температуры пласта и снижении вязкости нефти путём [4]:

- циклических тепловых обработок призабойных зон скважин,
- закачки в пласт теплоносителя (горячей воды, сухого или влажного водяного пара),
- инициирования в пласте процесса окисления и горения части нефти с закачкой в пласт воздуха или кислорода и, тем самым, создания подвижного очага окисления или горения нефти для выделения тепла.

Накопленный мировой опыт разработки залежей с высоковязкими нефтями доказал эффективность использования тепловых методов в таких случаях. Если традиционно применяемые технологии заводнения на месторождениях с нефтями повышенной и высокой вязкости могли обеспечить конечную нефтеотдачу не более 20–25%, то использование тепловых методов позволяет в ряде случаев довести нефтеотдачу до 40-45 % [9].

2. Изменение смачиваемости пород при нагреве пласта

Во многих работах отмечается возможность изменениях смачиваемости пород при нагреве, т.е. при воздействии высоких температур, которые при тепловых воздействиях разного рода могут на 100–200^{0}C превосходить начальную пластовую. Подобные изменения, в принципе, могут кардинально изменить характер фильтрации флюидов и результативность теплового воздействия.

К сожалению, приходится констатировать, что вопрос этот вообще пока слабо изучен. Более того, полученные разными авторами в различных экспериментах данные весьма противоречивы.

В обзоре [12] приводятся результаты ряда исследований изменения смачиваемости в зависимости от роста температуры при классических термических методах добычи (закачка пара, внутрипластовое горение и т.д.). В ряде экспериментов, как на насыпных песчаных моделях, так и на образцах песчаника, терригенная порода с ростом температуры становилась более гидрофильной. На насыпных моделях - остаточная нефтенасыщенность уменьшалась с ростом температуры, а на образцах песчаника об этом свидетельствовали также и изменения капиллярных кривых.

Однако в ряде экспериментов получены и прямо противоположные результаты – порода становилась более гидрофобной по мере повышения температуры, даже при закачке пара. В некоторых исследованиях, с ростом температуры сначала наблюдалось увеличение степени гидрофобности, а затем её уменьшение при последующем повышении температуры.

Более однозначны результаты, полученные в карбонатных породах, которые, согласно результатам практически всех исследователей, с повышением температуры становятся более гидрофильными.

Существует также мнение, что характер смачиваемости вообще никак не зависит от температуры, либо, при нагреве порода становится нейтральной по смачиваемости. Столь большой разброс результатов, вероятнее всего, определяется рядом объективных факторов. Во-первых, эксперименты проводились по различным методикам, с различными нефтями, и в разных условиях: на насыпных моделях, образцах керна, опытных участках. Влияние смачиваемости оценивалось также по-разному: по капиллярному впитыванию, по виду кривых относительных фазовых проницаемостей, и т.п. Во-вторых, смачиваемость зависит от многих физических факторов, по-разному реагирующих на повышение температуры. В их числе можно назвать: начальное водонасыщение и характер его изменения в процессе разработки; величина pH; состав нефти и возможность осаждения из неё асфальтенов; глинистость пород; устойчивость тонких плёнок воды на твёрдой поверхности и др. [12]. Результирующее изменение смачиваемости пород при нагреве будет зависеть от сочетания температурного поведения всех указанных факторов [12], имеющих в разных ситуациях разную значимость.

Отметим, что различие в характере изменения смачиваемости при нагреве часто отмечается и в других областях науки и техники.

Естественно, возникает вопрос, насколько сильно вероятное изменение смачиваемости пород при нагреве может повлиять на конечный результат применения тепловых методов добыч нефти, и, соответственно, насколько необходимо его учитывать при численном моделировании таких способов разработки. Для грубой качественной оценки значимости этого фактора были выполнен ряд модельных расчётов.

3. Оценка влияния изменения смачиваемости на показатели разработки

Модельный оценочный расчёт проводился с помощью математической модели двухфазной фильтрации несмешивающихся жидкостей (нефти и воды) Dz10 [13] в трёхмерной постановке для следующих значений параметров:

1. Количество ячеек по осям X, Y, Z 10-10-1.
2. Размер ячеек по осям X, Y, Z 50-50-20 м.
3. Размеры модели 500-500-20 м.
4. Пористость 0,2 д.ед.
5. Проницаемость 30 мД.
6. Объёмный коэффициент воды: 1,01 $м^3/м^3$.
7. Вязкость воды 0,3 мПа·с.
8. Сжимаемость воды $4{,}7\cdot10^{-5}$ 1/бар.
9. Объёмный коэффициент нефти 1,1 $м^3/м^3$.
10. Вязкость нефти 1 и 10 мПа·с.
11. Сжимаемость нефти $8{,}0\cdot10^{-5}$ 1/бар.
12. Растворимость газа в нефти 108 $м^3/м^3$.
13. Давление насыщения нефти газом 70 бар.
14. Сжимаемость пласта: $4{,}7\cdot10^{-5}$ 1/бар.
15. Плотность нефти при стандартных условиях 815 $кг/м^3$.
16. Плотность воды при стандартных условиях 1056 $кг/м^3$.
17. Глубина залегания 3390 м.
18. Начальное пластовое давление 339 бар.
19. Начальная водонасыщенность (гидрофильный пласт) 0,4 д.ед.
20. Начальная водонасыщенность (гидрофобный пласт) 0,2 д.ед.
21. Добывающая скважина находится в ячейке с координатами 10-10-1.
22. Нагнетательная скважина находится в ячейке с координатами 1-1-1.
23. Добывающая скважина работает при постоянном дебите жидкости 150 $м^3$/сут.
24. Нагнетательная скважина работает при постоянном забойном давлении 419 бар.
25. Срок разработки 3600 дней (примерно 10 лет).
26. Капиллярным давлением пренебрегаем, т.к. пласт однороден по пористости и проницаемости. Влияние капиллярных сил косвенно учитывается путём введения различных ОФП для случаев гидрофобного и гидрофильного коллектора.

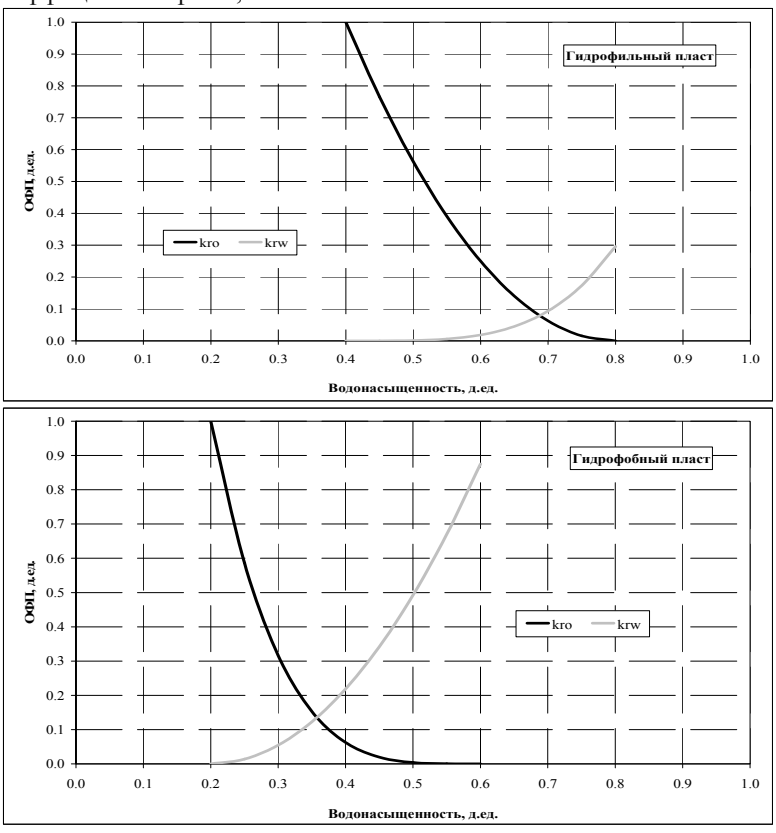

Рис.1. Принятые (по литературным данным) при расчёте кривые относительной фазовой проницаемости (ОФП) по нефти (kro) и воде (krw)

Ячейки разностной сетки прямоугольные, блочноцентрированные.

При расчётах были использованы типичные кривые относительных фазовых проницаемостей для гидрофильного и гидрофобного пласта, взятые по литературным данным и показанные на рис. 1.

Результаты расчётов приведены на рис. 2–5. Поскольку при нагреве нефтеносного пласта происходят два значимых эффекта – изменение вязкости нефти и характера смачиваемости, то на рисунках кривые, обусловленные влиянием обоих эффектов, сопоставлены со случаем сохранения начального характера смачиваемости, но уменьшения вязкости нефти.

Как видно из рис. 2 и 3, при нагреве гидрофильного пласта снижение вязкости нефти, естественно, улучшает показатель разработки, в частности, увеличивается период стабильного дебита нефти, а длительность падения дебита и роста обводнённости сокращается. В то же время, изменение поверхности породы с гидрофильной на гидрофобную сказывается на динамике разработки крайне отрицательно. Появление прорывов воды, проскальзывающей по фильтрационным каналам, поверхность которых «смазана» адсорбированными плёнками поверхностно-активных углеводородов, превышает положительный эффект от снижения вязкости нефти. Показатели разработки становятся даже хуже, чем при начальных параметрах пласта и нефти: падение дебита нефти начинается раньше и период обводнения становится более длительным. Накопленная добыча нефти значительно уменьшается.

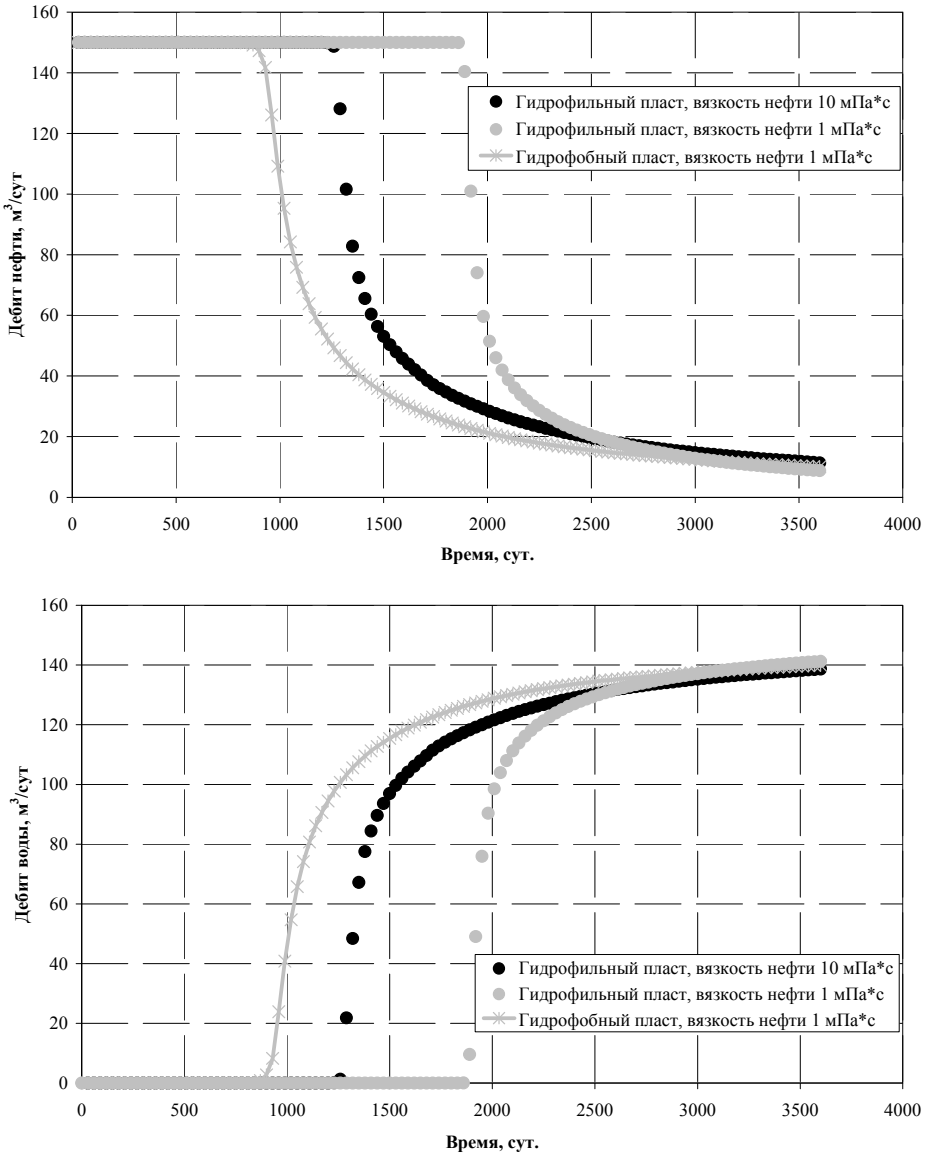

Рис. 2. Влияние изменения вязкости и характера смачиваемостина дебит воды и нефти при нагреве гидрофильного пласта

В гидрофобном пласте ситуация иная (рис. 4 и 5). В отсутствие нагрева процесс обводнения контролируется прорывами воды, которые обусловлены двумя факторами: вязкостной неустойчивостью вытеснения, поскольку вязкость нефти в 10 раз превышает вязкость воды; а также проскальзыванием воды по фильтрационным каналам, стенки которых покрыты адсорбированными углеводородами. Как показывают расчёты (рис. 4 и 5), в этом случае и уменьшение вязкости нефти, и изменение характера поверхности фильтрационных каналов – действую в одну сторону: уменьшают вероятность прорывов воды, увеличивают коэффициент вытеснения, снижают время начала падения дебита и длительность периода обводнения.

Накопленная добыча нефти значительно возрастает - за счёт обоих эффектов.

Следует подчеркнуть, что рассмотренный в статье случай во многом является иллюстративным. В действительности, при тепловых методах добычи поле температур в пласте неоднородно по пространству, поэтому характер смачиваемости поверхности также будет переменным по пространству. Это обстоятельство в статье не учитывалось.

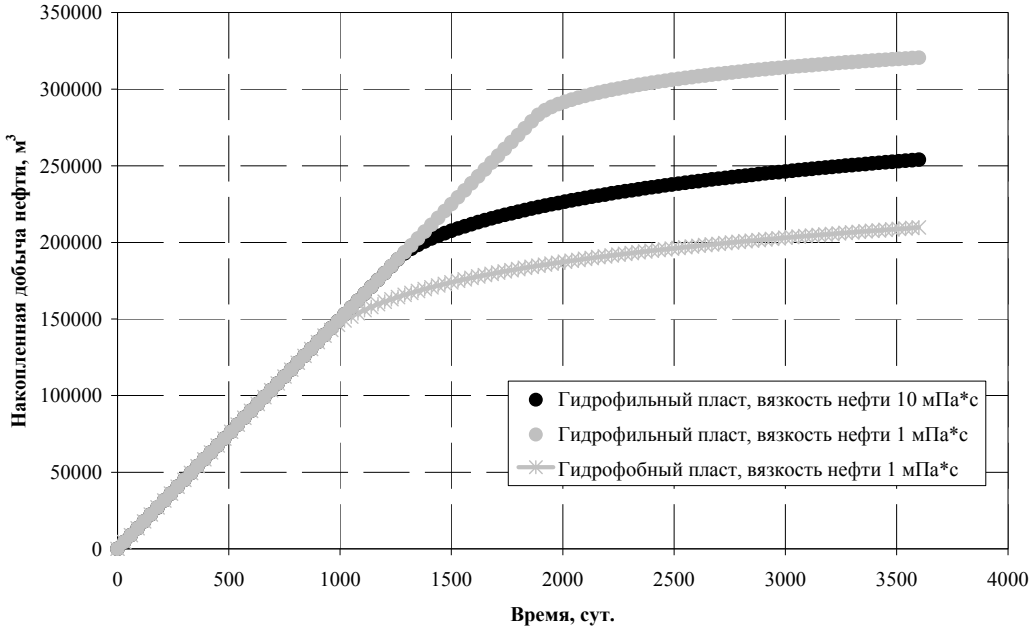

Рис. 3. Изменение накопленной добычи нефти при нагреве гидрофильного пласта

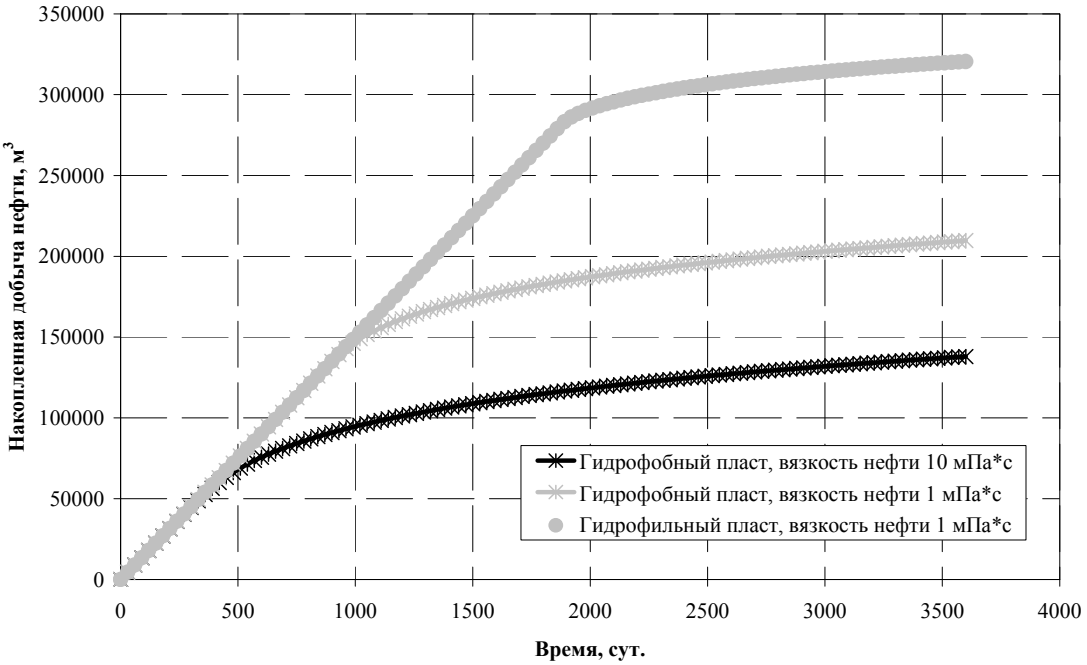

Рис. 4. Изменение накопленной добычи нефти при нагреве гидрофобного пласта

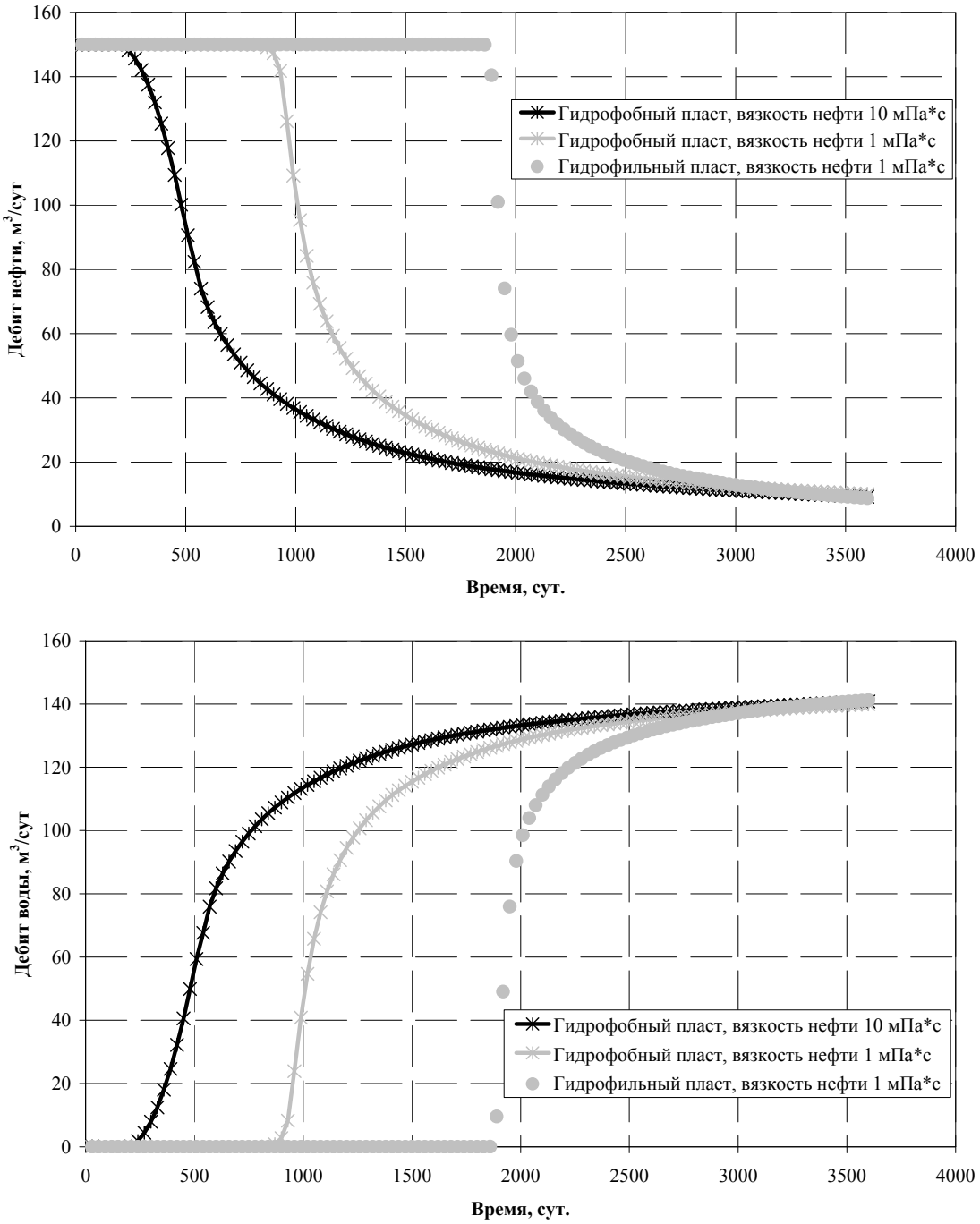

Рис.5. Влияние изменения вязкости и характера смачиваемости на дебит нефти и воды при нагреве гидрофобного пласта

Выводы

Несмотря на качественный, даже иллюстративный, характер выполненных расчётов, они, в совокупности с обзором литературных данных, свидетельствуют о том, что температурное изменение смачиваемости может оказывать значительное влияние на динамику добычи. При моделировании любых тепловых методов добычи нефти этот вопрос должен быть тщательно рассмотрен, что, к сожалению, как правило не выполняется на практике проектными организациями и НИИ.

При этом необходимо, по нашему мнению, выполнить цикл теоретических и экспериментальных работ по изучению изменения характера смачиваемости пород при нагреве, в широком интервале температур – от пластовых до, примерно, 400^0C. Такие исследования должны быть выполнены для широкого интервала параметров пласта и нефтей, охватывающего условия основных

объектов применения тепловых методов добычи нефти в нашей стране.

При моделировании тепловых методов на конкретном объекте необходимо, до начала моделирования, оценить значимость подобных эффектов в рассматриваемых условиях и необходимость их учёта. При положительном ответе – для их учёта система решаемых уравнений подземной гидродинамики должна быт соответствующим образом усложнена.

Публикация выполнена в рамках государственного задания на проведение фундаментальных научных исследований по программам РАН по теме (проекту) «1.33П. Разработка модели внутрипластового окисления углеводородов и неизотермических фильтрационных течений на основе суперкомпьютерного моделирования». (0065-2015-0112).

The necessity of taking into account rock wettabilty changes in thermal oil recovery methods simulation

V.A.Judin, I.V.Afanaskin

Abstract: Possibility of rock wettability changes as a result of using thermal methods of development is considered based on a brief literature review. Brief summary of experimentally obtained rocks wettability changes with temperature increase are presented. Carbonates become more water-wet with temperature increase while experimental results for terrigenous formations are quite ambiguous. On the base of simple computer modelling it is shown that the production history might be improved by wettability changes (in oil-wet rocks) or made worse (in water-wet rocks).A thorough investigation of the problem is proposed under thermobaric conditions of Russian oil fields with viscous oils on which thermal methods are planned to be applied. This is necessary to increase reservoir modeling reliability.

Key words: thermal methods of oil development, wettability, improved oil recovery, reservoir simulation.

Литература

1. А.Орёл. Разработка трудноизвлекаемых запасов связана с инвестиционными рисками. «Нефть и жизнь», т. 108 (2016), № 8.
2. А.Н.Ищенко. О перспективах развития ТЭК России. «Недропользование - XXI век», 2017, №1, 12–13.
3. BP Energy Outlook - 2017. Режим доступа в сети интернет: http://www.imemo.ru/files/File/ru/conf/2017/07022017/07022017-PRZ-EO17-Presentation-Spencer%20short.pdf
4. В.Б.Бетелин, В.А.Юдин, И.В.Афанаскин, С.Г.Вольпин, Р.М.Кац, А.В.Королёв. Создание отечественного термогидросимулятора – необходимый этап освоения нетрадиционных залежей углеводородов России. М., ФГУ ФНЦ НИИСИ РАН, 2015.
5. Б.В.Успенский. Научно-методические основы поиска, разведки и освоения природных битумов. Дисс. на соиск. уч. степ. докт. геол.-мин. наук. Казань, 2005.
6. И.В.Николин. Методы разработки тяжёлых нефтей и природных битумов. «Наука – фундамент решения технологических проблем развития России», 2007, № 2, 58–64.
7. В.П.Якуцени, Ю.Э.Петрова, А.А.Суханов. Нетрадиционные ресурсы углеводородов - резерв для восполнения сырьевой базы нефти и газа в России. «Нефтегазовая геология. Теория и практика», т. 1 (2009), № 4.
8. Д.Ю.Чиркова, Н.А.Красноярова, И.Г.Ященко. Проблемы рационального использования вязких и тяжелых нефтей российской Арктики. «XVI Международная научно-практическая конференция имени профессора Л.П. Кулёва». Томск, Томский политехнический университет, 2015.
9. Я.Бао, Ю.Тянь, А.В.Сиднев. Глинистые низкопроницаемые коллекторы в нефтегазоносных бассейнах Китая и технологии добычи нефти. «Современные наукоемкие технологии», 2008, № 2, 57–58.

10. В.Е.Борисов. Композиционная неизотермическая модель фильтрации в пористой среде с учетом химических реакций и активной твердой фазы. «Препринты ИПМ им. М.В.Келдыша», 2013, № 91.
11. Д.Г.Антониади, А.Р.Гарушев, Б.Г.Ишханов. Настольная книга по термическим □методам □добычи □нефти. Краснодар: «Советская Кубань», 2000.
12. A.Punase, A.Zou, R.Elputranto. How Do Thermal Recovery Methods Affect Wettability Alteration? «Journal of Petroleum Engineering», V. 2014, Article ID 538021.
13. 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.

К теории интерполирования в пространстве гладких чебышёвских обобщённо-полиномиальных сплайнов

В.Б. Демидович

ФГБОУ ВО « Московский государственный университет имени М.В.Ломоносова», Механико-математический факультет и ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail : vasdem@mech.math.msu.su

Аннотация: В пространстве гладких чебышёвских обобщённо-полиномиальных сплайнов вводятся специальные сплайны для эффективного представления соответствующих интерполяционных сплайнов и оценок их погрешностей.

Ключевые слова: гладкие чебышёвские системы, обобщённые полиномы, обобщённо-полиномиальные сплайны, интерполяция, оценка погрешности.

В библиографии книг [А-1 – А-13] и журнальных статей [Б-1 – Б-6] рассматриваются различные вопросы теории (*непрерывных* и *гладких*) чебышёвских обобщённых полиномом и обобщённо-полиномиальных сплайнов. В частности, при исследовании *интерполяционных свойств* гладких чебышёвских обобщённых ***полиномов***, в [Б-5], ***во-первых***, отмечена *«безусловная» однозначная разрешимость как простой* интерполяционной задачи *(для любой непрерывной функции),* так и *кратной* интерполяционной задачи (*для любой функции соответствующей гладкости*), а, ***во-вторых***, указаны два *эффективных* представления (*лагранжевского* типа и *ньютоновского* типа) как для самого интерполяционного чебышёвского обобщённого полинома, так и для его погрешности, в задаче *простой* интерполяции, и, соответственно, два эффективных представления *(лагранжа-эрмитовского* типа и *ньютоно-эрмитовского* типа) как для самого *кратно*-интерполяционного чебышёвского обобщённого полинома, так и для его погрешности, в задаче *кратной* интерполяции. В свою очередь, при исследовании *интерполяционных свойств* гладких чебышёвских обобщённо–полиномиальных ***сплайнов***, в [Б-6], ***во-первых***, сформулирован *критерий разрешимости* (в том числе и *однозначной разрешимости) для любой функции заданной гладкости в кратной (а, формально, и простой) интерполяционной задаче*, и, ***во-вторых***, указано *“единое” теоретическое представление* (не претендующее на свою практическую эффективность) для интерполяционного чебышёвского обобщённо-полиномиального сплайна *в таких задачах.*

Углубляя эти исследования, в данной работе для задачи интерполирования в пространстве гладких чебышёвских обобщённо-полиномиальных сплайнов вводятся соответствующие *базисные* сплайны для *эффективного представления самих интерполяционных сплайнов*, а также некоторые *специальные* сплайны в связи с оценкой их погрешностей.

I. Исходные сведения

1. Напомним, что совокупность функций

$$\{e_k(\cdot)\}_{k=0}^{\infty} \in C^{(\infty)}([a,b]) \qquad (1.1)$$

образует, на заданном отрезке $[a,b] \subset R \ \ (a<b)$, *фундаментальную гладкую* ***чебышёвскую систему*** *в пространстве* $C([a,b])$ если:

А) для каждого её (конечного) набора функций

$$\{e_k(\cdot)\}_{k=0}^{n} \quad (n = 0, 1, 2, ...) \qquad (1.2)$$

построенная по нему *нетривиальная линейная комбинация “ порядка n”* с постоянными коэффициентами

$$\phi_n(t) = \sum_{i=0}^{n} c_i e_i(t) \quad (\sum_{i=0}^{n} c_i^{\,2} \neq 0) \qquad (1.3)$$

имеет на $[a,b]$ не более n корней *с учётом их алгебраических кратностей* (то есть, согласно принятой терминологией в теории «чебышёвских систем», каждый набор функций (1.2) служит *ECT – системой n-го порядка на отрезке [a, b]* от сокращения английского названия *Extended Complete Tchebycheff systeme*);

Б) *линейное пространство*

$$\Phi([a,b]) = \bigcup_{n=0}^{\infty} \Phi_n([a,b]), \qquad (1.4)$$

где

$$\Phi_n([a,b]) = span\{e_i(t)\}_0^n \qquad (1.5)$$

- линейная оболочка набора (1.2), порождающая в пространстве $\Phi([a,b])$ подпространство, для размерности которого, очевидно, справедливо

$$\dim \Phi_n([a,b]) = (n+1), \qquad (1.6)$$

плотно в пространстве $C([a,b])$ относительно его *равномерной нормировки*, и, тем самым, для *равномерного замыкания* пространства $\Phi([a,b])$ имеет место равенство

$$cl\ \Phi([a,b]) = C([a,b]). \qquad (1.7)$$

Всякий элемент из $\Phi_n([a,b])$ мы, естественно, будем называть *гладким* **чебышёвским обобщённым полиномом.**

В [Б-3 - Б-5] приводятся различные примеры подобных *чебышёвских систем* и соответствующих *обобщённых полиномов*. Там же предложен *аппарат* для их исследования.

2. В свою очередь, напомним, что на отрезке $[a,b]$ с *фиксированным разбиением*

$$\Omega_N[a,b]:\ a = t_0 < t_1 < ... < t_N < t_{N+1} = b, \qquad (1.8)$$

образованным набором внутренних узлов $\{t_j\}_{j=1}^N$ $(N \geq 1 - целое)$ данного отрезка и его концами, функцию $S_n(t)$ мы называем *гладким чебышёвским обобщённо-полиномиальным сплайном* (*порядка* $n \geq 0$ /$n \geq 0 - целое$/, с *гладкостями* $\{\nu_j\}_1^N$ /$-1 \leq \nu_j < n$ *– целые* / в узлах $\{t_j\}_1^N$ разбиения $\Omega_N[a,b]$), если:

А) на каждом “подотрезке”

$$[t_j, t_{j+1}] \quad (j = 0, 1, ..., N) \qquad (1.9)$$

$S_n(t)$ является гладким чебышёвским обобщённым полиномом из $\Phi_n([a,b])$, причём хотя бы на одном под-отрезке соответствующий обобщённый полином имеет в точности *n-ый* порядок;

Б) во внутренних узлах разбиения $\Omega_N[a,b]$

$$t_j \quad (j = 1, ..., N) \qquad (1.10)$$

функции $S_n(t),\ S_n^{(1)}(t),\ ...,S_n^{(\nu_j)}(t)$ *непрерывны*, а $S_n^{(\nu_j+1)}(t)$ *терпит конечный разрыв.*

Числом

$$\nu =: \min_{1 \leq j \leq N} \nu_j \qquad (-1 \leq \nu < n) \qquad (1.11)$$

тогда определяется *гладкость* данного сплайна *на всём отрезке* $[a,b]$.

Замечание 1.

В теории чебышёвских обобщённо-полиномиальных сплайнов на заданном разбиении $\Omega_N[a,b])$, наряду со *сплайнами порядка* n (с *гладкостями* $\{\nu_j\}_1^N$ /$\nu_j < n$/ в узлах $\{t_j\}_1^N$), иногда возникает необходимость рассматривать, также, *сплайны порядка* $n+1$ (с *гладкостями* $\{\nu_j\}_1^N$ /$\nu_j < n+1$/ в узлах $\{t_j\}_1^N$), причём полагается, что для такого сплайна $S_{n+1}(t)$ справедливо:

во-первых, на каждом “под-отрезке” $[t_j, t_{j+1}]$ $(j = 0, 1, ..., N)$ он является гладким чебышёвским обобщённым полиномом из $\Phi_{n+1}([a,b])$, причём хотя бы на одном под-отрезке соответствующий обобщённый полином имеет в точности $(n+1) - ый$ порядок;

и, *во-вторых*, во внутренних узлах t_j $(j = 1, ..., N)$ разбиения $\Omega_N[a,b]$ *функции* $S_{n+1}(t),\ S_{n+1}^{(1)}(t),\ ...,S_{n+1}^{(\nu_j)}(t)$ *непрерывны*, а $S_n^{(\nu_j+1)}(t)$ *терпит конечный разрыв.*

Понятно, что тогда числом $\nu =: \min_{1 \leq j \leq N} \nu_j$ $(-1 \leq \nu < n+1)$ определяется *гладкость* данного сплайна *на искомом отрезке* $[a,b]$.

В [Б-6] поясняется, что совокупность *гладких чебышёвских обобщённо-полиномиальных сплайнов порядка* $n \geq 0$ *с гладкостями* $\{\nu_j\}_1^N$ в узлах $\{t_j\}_1^N$ разбиения (1.8) порождает (линейное) *пространство гладких чебышёвских обобщённо-полиномиальных сплайнов* (*порядка n глад-кости* ν), обозначаемое через $S_n^{\nu}(\Omega_N[a,b])$. При этом, как показано в [Б-6], *значение размерности* пространства $S_n^{\nu}(\Omega_N[a,b])$ определяется выражением

$$\dim S_n^{\nu}(\Omega_N[a,b]) = (n+1) + N(n-\nu)_+, \qquad (1.12)$$

где

$$(n-\nu)_+ = (n-\nu)^+ = \begin{cases} n-\nu & при\ n > \nu \\ 0 & при\ n \leq \nu \end{cases} \qquad (1.13)$$

- соответствующая *срезка*.

Всякий элемент $S_n(t)$ из $S^{\nu}_{n}(\Omega_N[a,b])$ мы, будем называть просто *гладким* **чебышёвским обобщённо-полиномиальным сплайном.**

В [Б-6] вводятся и некоторые *характеристики* для анализа подобных обобщённо-полиномиальных сплайнов, связанные с *теорией интерполирования функций.*

В частности, для любого гладкого чебышёвского обобщённо-полиномиального сплайна $S_n(t) \in S^{\nu}_{n}(\Omega_N[a,b])$ определено понятие его *суммарного порядка* $m(S_n)$ выражением

$$m(S_n) =: n + \sum_{j=1}^{N} (n-\nu_j)_+ . \qquad (1.14)$$

В свою очередь, для самого *пространства сплайнов* $S^{\nu}_{n}(\Omega_N[a,b])$ под его *суммарным порядком* m мы будем понимать величину

$$m = n + N\left(n-\nu\right)_+ . \qquad (1.15)$$

Ясно, что всегда

$$m(S_n) \le m = \dim S^{\nu}_{n}(\Omega_N[a,b]) - 1. \qquad (1.16)$$

II. Об эффективном представлении интерполяционного сплайна в пространстве гладких чебышёвских обобщённо-полиномиальных сплайнов

При описании *эффективного представления* интерполяционных сплайнов (вообще говоря, *кратных*, а в частных случаях *простых*) в пространстве $S^{\nu}_{n}(\Omega_N[a,b])$, определяемых интерполяционной задачей, когда для функции

$$x(t) \in C^{(n)}([a,b]) \qquad (2.1)$$

в заданном наборе узлов

$$\{\xi_0 < \ldots < \xi_p\} \in [a,b] \qquad (2.2)$$

с кратностями

$$\{n \ge q_k \ge 0\}_{k=0}^{p} \qquad (2.3)$$

нужно построить гладкий чебышёвский обобщённо-полиномиальный сплайн $S_n(t;\, x)$ в пространстве

$$S^{\nu}_{n}(\Omega_N[a,b])$$

$$(\dim \{S^{\nu}_{n}(\Omega_N[a,b])\} = \sum_{k=0}^{p}(q_k+1) = m+1), \qquad (2.4)$$

с реализацией равенств

$$S_n^{(l)}(\xi_k;\, x) = x^{(l)}(\xi_k)$$
$$(k = 0, 1, 2, \ldots, p; \quad l = 0, 1, \ldots, q_k), \qquad (2.5)$$

важную роль играют соответствующие *базисные* гладкие чебышёвские обобщённо-полиномиальные *сплайны*, аналогичные *базисным* гладким чебышёвским обобщённым *полиномам*. При этом здесь используется лишь *Лагранжева-Эрмитова* (в частности, просто *Лагранжева*) концепция, поскольку *Ньютоновско-Эрмитовый* (или же просто *Ньютоновский*) *подход* (основанный на соответствующих «разделённых разностях» - про *обобщённые* разделённые разности см. [Б-5]) *для сплайн-интерполяции*, как нам представляется, *"не имеет практических перспектив"* (из-за неизбежного возникновения, при его разработке, "проблем с сингулярностью").

Явный вид системы базисных гладких чебышёвских обобщённо-полиномиальных *сплайнов Лагранжева-Эрмитова типа*

$$\{S_{ij}(t)\} \in S^{\nu}_{n}(\Omega_N[a,b]) \ (i=\overline{0,p};\ j=\overline{0,q_i};$$
$$\sum_{i=0}^{p}(q_i+1) = m+1;\ m = n + N(n-\nu)_+) \qquad (2.6)$$

в общей ситуации (по отношению к узлам интерполяции и разбиения $\Omega_N[a,b]$) "разумным образом" не определяется. Однако в каждой конкретной ситуации такая *система базисных сплайнов* может быть однозначным образом построена из "естественных" (для неё) условий вида

$$S_{ij}^{(l)}(\xi_k) = \delta_i^k \cdot \delta_j^l$$
$$(i,k = \overline{0,p};\ j = \overline{0,q_i};\ l = \overline{0,q_k}), \qquad (2.7)$$

если только узлы интерполяции $\{\xi_k\}_{k=0}^{p}$ и узлы разбиения $\{t_k\}_{k=0}^{N+1}$ расположены согласно *критерию однозначной разрешимости* исходной *интерполяционной задачи*, то есть если эти узлы удовлетворяют неравенствам

$$\Gamma[t_i,\, t_j] \le (n+1) + (j-i-1)_+(n-\nu)_+$$
$$(0 \le i \le j \le N+1), \qquad (2.8)$$

где $\Gamma[t_i,\, t_j]$ - *счётчики узлов интерполирования на отрезке* $[t_i,\, t_j]$, характеризующие количество узлов интерполяции (с учётом их кратностей), попадающих в *промежуток* $[t_i,\, t_j]$ (см. *Теорему 3* в [Б-6]).

(*Примечание 1.* Здесь используется *символ Кронекера* δ_α^β, определяющий функцию от двух *целых* переменных α и β,

равную *1* , если $\alpha = \beta$, и *0* при $\alpha \neq \beta$. Немецкий учёный Леопольд Кронекер /Leopold Kronecker/ (1823-1891) ввёл в математическую литературу этот символ в 1866-ом году.)

Очевидно, что удовлетворяющие соотношениям (2.7) совокупность гладких чебышёвских обобщённо-полиномиальных сплайнов из пространства $S_n^{\nu}(\Omega_N[a,b])$ в случае, когда их количество равно размерности данного пространства, будут составлять *базис* этого пространства. Тем самым *интерполяционный сплайн* $S_n(t;\, x)$ для функции $x(t)$ может быть записан с помощью (структурной) формулы

$$S_n(t;\, x) = \sum_{i=0}^{p}\sum_{j=0}^{q_i} S_{ij}(t)\cdot x^{(j)}(\xi_i), \qquad (2.9)$$

являющейся "аналогом" *Лагранжева-Эрмитовой формы записи* соответствующего *интерполяционного* гладкого чебышёвского обобщённого *полинома*. Достоинством данной формы записи служит тот факт, что при переходе к интерполированию новой функции в рассматриваемых интерполяционных условиях не будет необходимости заново строить совокупность базисных гладких чебышёвских обобщённо-полиномиальных сплайнов из пространства $S_n^{\nu}(\Omega_N[a,b])$, а к её недостаткам следует отнести неизбежность нового построения такой совокупности при малейшем изменении узлов интерполяции в рассматриваемой интерполяционной задаче.

(*Примечание 2.* Для *классических* полиномиальных сплайнов формула вида (2.9), при условиях (2.7), широко известна (для Лагранжева-Эрмитовой ситуации см., например, [А-5, с. 40-42], а для собственно Лагранжевой ситуации см., в частности, [А-7, с. 28-29]).)

II. О специальных гладких чебышёвских обобщённо-полиномиальных сплайнах

К другому типу гладких чебышёвских обобщённо-полиномиальных сплайнов в пространстве $S_n^{\nu}(\Omega_N[a,b])$ можно отнести сплайны, в некотором смысле напоминающие "наиболее извивающиеся обобщённо-полиномиальные ужи в разделимом коридоре" (подробнее про "ужи" см. [А-12, с.77-85]), строящиеся по узлам $\{t_j\}_{j=1}^{N}$ исходного разбиения $\Omega_N[a,b]$.

Именно, всякий сплайн $S_n(t)$ из пространства $S_n^{\nu}(\Omega_N[a,b])$ можно охарактеризовать поведением его *обобщённой производной n-го порядка в силу исходной чебышёвской системы*, определяемой соотношением

$$D_n[S_n(t)] =: \frac{W(e_0(t),\ \ldots\ ,e_{n-1}(t),\ S_n(t))}{W(e_0(t),\ \ldots,\ e_{n-1}(t))}, \quad D_0[S_n(t)] =: S_n(t) \quad (n = 1,\ 2,\ \ldots), \qquad (3.1)$$

где

$$W(e_0(t),\ e_1(t),\ \ldots,\ e_{n-1}(t),\ S_n(t)) = \begin{Vmatrix} e_0(t) & \ldots & e_{n-1}(t) & e_n(t) \\ e_0^{(1)}(t) & \ldots & e_{n-1}^{(1)}(t) & e_n^{(1)}(t) \\ \ldots & \ldots & \ldots & \ldots \\ e_0^{(n-1)}(t) & \ldots & e_{n-1}^{(n-1)}(t) & e_n^{(n-1)}(t) \\ S_n^{(n)}(t) & \ldots & S_n^{(n)}(t) & S_n^{(n)}(t) \end{Vmatrix} \quad (n = 1,\ 2,\ldots) \qquad (3.2)$$

и

$$W(e_0(t),\ e_1(t),\ \ldots,\ e_{n-1}(t)) = \begin{Vmatrix} e_0(t) & \ldots & e_{n-1}(t) \\ e_0^{(1)}(t) & \ldots & e_{n-1}^{(1)}(t) \\ \ldots & \ldots & \ldots \\ e_0^{(n-1)}(t) & \ldots & e_{n-1}^{(n-1)}(t) \end{Vmatrix} \quad (n = 1,\ 2,\ldots) \qquad (3.3)$$

- детерминанты вводимых производных (подробнее про такие *обобщённые производные* см. [А-12, с. 24-27]). Отсюда выделяется совокупность сплайнов $\{S_n(t)\}$, у которых все их *гладкости* $\{\nu_j = \nu(S_n;\, t_j)\}_{j=1}^{N}$ в узлах $\{t_j\}_{j=1}^{N}$ разбиения $\Omega_N[a,b]$ *одинаковы* и *равны*

$n-1$, а обобщённые производные $D_n[S_n(t)]$ между искомыми узлами удовлетворяют (своего рода "*альтернансным*") требованиям:

$$|D_n[S_n(t)]| \ge 1 \quad (t \in (t_j; t_{j+1}); \ j = \overline{0, N}); \tag{3.4'}$$

$$\operatorname{sgn} D_n[S_n(t)]|_{t\in(t_{j-1},t_j)} = -\operatorname{sgn} D_n[S_n(t)]|_{t\in(t_j,t_{j+1})} \ (j = \overline{1, N}). \tag{3.4"}$$

В этом случае $S_n(t)$ мы будем называть *квази-перфектным гладким чебышёвским обобщённо-полиномиальным сплайном n-го порядка* и, соответственно, просто *перфектным гладким чебышёвским обобщённо-полиномиальным сплайном n-го порядка*, если соотношения (3.4') реализуются в виде *строгих равенств.* При этом, для определённости, сплайн $S_n(t)$ мы будем называть *верхним* "*ужом*", если

$$\operatorname{sgn} D_n[S_n(t)]|_{t\in(t_0,t_1)} = +1, \tag{3.5'}$$

и *нижним "ужом"*, если

$$\operatorname{sgn} D_n[S_n(t)]|_{t\in(t_0,t_1)} = -1. \tag{3.5"}$$

Замечание 2.

В "*классической* полиномиальной ситуации" термин "квази-перфектность" не вводится, поскольку производные *n-го порядка* от алгебраического многочлена "*реально n-ой степени"* есть просто *константа*, и потому добиться реализации строгих равенств в соотношениях типа (3.4') конструктивно не представляется затруднительным делом.

В то же время, в "*общей* чебышёвской ситуации" (естественно, за исключением оговоренной классической ситуации), напротив, говорить о возможности реализовать требуемые равенства весьма проблематично, и потому тогда основным для нас будет термин "квази-перфектность".

Добавим, что в теории классических полиномиальных сплайнов, наряду с прилагательным "перфектный", к сплайнам широко используются прилагательные-синонимы "совершенный" или же "идеальный".

В отношении *явного вида* квази-перфектного гладкого чебышёвского обобщённо-полиномиального сплайна n-го порядка имеет место следующее утверждение:

Теорема 1. Всякий квази-перфектный гладкий чебышёвский обобщённо-полиномиальный сплайн n-го порядка $S_n(t)$ пространства $S^{\nu}_n(\Omega_N[a,b])$, с гладкостями $\{\nu_j = \nu(S_n; t_j)\}_{j=1}^N$ в узлах $\{t_j\}_{j=1}^N$ разбиения $\Omega_N[a,b]$, представим в виде

$$S_n(t) = \pm\frac{G_n(t)}{\underset{\sim}{\theta}_n(t)} + \sum_{i=0}^{n-1} c_i K_i(t;a), \tag{3.6}$$

где

$$G_n(t) = \{K_n(t;a) + 2\sum_{j=1}^{N}(-1)^j \theta_n^j K_n^+(t;t_j)\} \tag{3.7}$$

- функция, далее называемая *альтернансной образующей* n –го порядка,

$$\widehat{\theta}_n = \max_t |D_n[e_n(t)]|,$$

$$\underset{\sim}{\theta}_n = \min_t |D_n[e_n(t)]|, \quad \theta_n = \frac{\widehat{\theta}_n}{\underset{\sim}{\theta}_n} \ge 1 \tag{3.8}$$

- чебышёвские обобщённо-интерполяционные *константы* (введенные в [А-13, с. 55]),

$$K_i(t; t_j) = \frac{W\begin{pmatrix} e_0, & \dots & , e_{i-1}, & e_i \\ t_j, & \dots & , t_j, & t \end{pmatrix}}{W(e_0(t_j), ..., e_{i-1}(t_j))},$$

$$K_0(t; t_j) =: e_0(t_j) \quad (i = \overline{0, n}; \ j = \overline{0, N}) \tag{3.9}$$

- простейшие обобщённо-полиномиальные ядра, строящиеся с помощью определителей вида (3.2)-(3.3),

$$K_n^+(t;t_j) \quad (j = \overline{1,N}) \tag{3.10}$$

- соответствующие *срезки*, а

$$\{c_i\}_{i=0}^{n-1} \tag{3.11}$$

- числовые коэффициенты.

Краткое пояснение к проведению доказательства.

Доказательство этого утверждения можно провести «непосредственной проверкой» *выполнения* для обобщённой производной $D_n[S_n(t)]$ от функции $S_n(t)$, задаваемой формулами (3.6)-(3.7), *требований* (3.4')-(3.4"), при переходе от интервала (t_0, t_1) к интервалу (t_1, t_2), затем от интервала (t_1, t_2) к интервалу (t_2, t_3) и так далее. При этом станет ясно, что знаком "+" будет характеризоваться сплайн $S_n(t)$, являющийся *верхним ужом*, а знаком "-" - сплайн $S_n(t)$, являющийся *нижним ужом*

Замечание 3.

Если в формуле (3.6) положить $\{c_i = 0\}_{i=0}^{n-1}$, то получим пару *нормализованных* квази-перфектных гладких чебышёв-

ских обобщённо-полиномиальных сплайнов-ужей *n-го порядка*

$$S_n(t) = \pm \frac{G_n(t)}{\underset{\sim}{\theta}_n(t)} =$$

$$\pm \frac{1}{\underset{\sim}{\theta}_n(t)} \{K_n(t;a) + 2\sum_{j=1}^{N} (-1)^j \theta_n^j K_n^+(t;t_j)\}. \quad (3.12)$$

Замечание 4.

В классической полиномиальной ситуации справедливо

$$\widehat{\theta}_n = \underset{\sim}{\theta}_n = n!, \qquad \theta_n = \frac{\widehat{\theta}_n}{\underset{\sim}{\theta}_n} = 1, \quad (3.13)$$

и потому соотношения (3.6)-(3.7) можно записать в виде единой формулы

$$S_n(t) = \pm \frac{1}{n!} \{(t-a)^n + 2\sum_{j=1}^{N} (-1)^j (t-t_j)_+^n\} + \sum_{i=0}^{n-1} c_i (t-a)^i, \quad (3.14)$$

определяющей нижние и верхние перфектные сплайн-ужи n –го порядка. Соответственно, пара классических "нормализованных" перфектных сплайн-ужей n-го порядка тогда описывается формулой

$$S_n(t) = \pm \frac{1}{n!} \{(t-a)^n + 2\sum_{j=1}^{N} (-1)^j (t-t_j)_+^n\}. \quad (3.15)$$

Ясно, что *n-ые* производные таких сплайнов будут удовлетворять соотношению

$$S_n^{(n)}(t) = \pm \{1 + 2\sum_{j=1}^{N} (-1)^j (t-t_j)_+^0\} \quad (3.16)$$

$$(t \in (t_j, t_{j+1}), \quad j = \overline{0, N}),$$

и тем самым $S_n^{(n)}(t)$ тогда является просто *кусочно-постоянной функцией*, принимающей на соседних промежутках разбиения значения "± 1", причём в каждом узле $\{t_j\}_{j=1}^N$ разбиения $\Omega_N[a,b]$ меняющей свой знак. Отсюда, в частности, следует, что в этом случае $S_n^{(n-1)}(t)$ будет *непрерывной ломаной функцией*, угловые коэффициенты звеньев которой *по абсолютной величине равны единице* и в каждой своей вершине $\{t_j\}_{j=1}^N$ меняющей знак.

(*Примечание 3.* Свойства классических перфектных сплайнов активно исследовались (см., например, [А-11, с. 160 и с. 191], [А-4, с. 71 и с. 75], [А-5, с. 8 и с. 12],), в то время как анализ квази-перфектных гладких чебышёвских обобщённо-полиномиальных сплайнов является пока лишь "общей перспективой".)

IV. Об оценке погрешности интерполяции функций гладкими чебышёвскими обобщённо-полиномиальными сплайнами *n-го* порядка на классе функций $W^{(n+1)}[M_{n+1}; a,b]$

Продемонстрируем теперь, как можно использовать введенные нами квази-перфектные гладкие чебышёвские обобщённо-полиномиальные сплайны для оценки погрешности интерполяции в пространстве сплайнов n-го порядка $S_n^{\nu}(\Omega_N[a,b])$. При этом, учитывая сказанное в *пункте II*, мы будем иметь ввиду лишь Лагранжева-Эрмитову концепцию интерполирования, причём полагать, что интерполируемая функция принадлежит классу функций

$$W^{(n+1)}[M_{n+1}; a,b], \quad (4.1)$$

состоящему из совокупности функций $\{x(t)\} \subset C^{(n+1)}([a,b])$, удовлетворяющих оценке

$$\max_t |D_{n+1}[x(t)]| \le M_{n+1}. \quad (4.2)$$

Итак, пусть для функции

$$x(t) \in W^{(n+1)}[M_{n+1}; a,b] \quad (4.3)$$

рассматривается *интерполяционная задача*, когда для заданного набора узлов интерполяции

$$\{\xi_0 < \ldots < \xi_p\} \in [a,b] \quad (4.4)$$

с кратностями

$$\{n \ge q_k \ge 0\}_{k=0}^p \quad (4.5)$$

строится в пространстве $S_n^{\nu}(\Omega_N[a,b])$ гладкий чебышёвский обобщённо-полиномиальный сплайн $S_n(t; x)$ с реализацией равенств

$$S_n^{(l)}(\xi_k; x) = x^{(l)}(\xi_k)$$
$$(k = 0, 1, 2, \ldots, p; \quad l = 0, 1, \ldots, q_k), \quad (4.6)$$

причём полагается, что узлы интерполяции $\{\xi_k\}_{k=0}^p$ и узлы исходного разбиения $\{t_k\}_{k=0}^{N+1}$ отрезка $[a,b]$ удовлетворяют *критерию однозначной разрешимости* поставленной *интерполяционной задачи* (см. (2.8)). Тогда в отношении погрешности интерполяции

$$R(t;x) =: x(t) - S_n(t;x) \quad (4.7)$$

справедлив следующий результат:

Теорема 2. В интерполяционной задаче (4.3)-(4.6) в отношении погрешности (4.7) справедлива *структурная оценка*

$$\|R(t;\,x)\|_{C([a,b])} \le \frac{M_{n+1}}{\underset{\sim}{\theta}_{n+1}} \|F_{n+1}(t)\|_{C([a,b])}, \tag{4.8}$$

где

$$\underset{\sim}{\theta}_{n+1} = \min_{a\le t\le b} |\,D_{n+1}[e_{n+1}(t)]\,| > 0, \tag{4.9}$$

а

$$F_{n+1}(t) = F_{n+1}(t;\ \Phi([a,b]),\ \Omega_N([a,b]),\ \nu,\ \{\xi_k\}_{k=0}^{p},\ \{q_k\}_{k=0}^{p}) \tag{4.10}$$

- не зависящая от $x(t)$ функция, аннулирующаяся (кратным образом) в узлах интерполяции

$$F_{n+1}^{(l)}(\xi_k) = 0 \ \ (k=\overline{0,p};\ \ l=\overline{0,q_k}), \tag{4.11}$$

структура которой может быть задана формулой

$$F_{n+1}(t) = K_{n+1}(t;a) + \sum_{i=0}^{n} \alpha_i\, K_i(t;\,a) + \sum_{j=1}^{N} \{2\,(-1)^j \cdot \theta_{n+1}^{j} \cdot K_{n+1}^{+}(t;t_j) + \sum_{i=\nu+1}^{n} \beta_{ij}\, K_i^{+}(t;t_j)\}, \tag{4.12}$$

в которой

$$\{\alpha_i = \alpha_i(G_{n+1})\}_{i=0}^{n},\ \ \{\beta_{ij} = \beta_{ij}(G_{n+1})\}_{i=\nu+1,\ j=1}^{n,\ N} \tag{4.13}$$

- числовые коэффициенты, определяемые альтернансной образующей $(n+1)-го$ порядка

$$G_{n+1}(t) = K_{n+1}(t;a) + 2\sum_{j=1}^{N} (-1)^j\, \theta_{n+1}^{j} K_{n+1}^{+}(t;t_j). \tag{4.14}$$

Эскиз доказательства.

Поскольку у входящих в пространство $S_n^{\nu}(\Omega_N[a,b])$ *квази-перфектных чебышёвских обобщённо-полиномиальных сплайнов n-го порядка* для *гладкости* справедливо равенство

$$\nu = n-1, \tag{4.15}$$

то *суммарный порядок* m этого пространства определяется величиной

$$m = n + N(n-\nu)_+ = n + N. \tag{4.16}$$

Непосредственной проверкой покажем, что, в условиях (4.15)-(4.16), в качестве функции $F_{n+1}(t)$ можно взять разность

$$G_{n+1}(t) - S_n(t;\,G_{n+1}) =: R(t;\,G_{n+1}), \tag{4.17}$$

где $G_{n+1}(t)$ - альтернансная образующая $(n+1)-го$ порядка, определяемая формулой (4.14), а

$$S_n(t;G_{n+1}) \in S_n^{\nu}(\Omega_N[a,b]) \tag{4.18}$$

- отвечающий задаче (4.3)-(4.7) интерполяционный сплайн этой образующей.

Именно, рассуждая "от противного", положим, что в некоторой точке

$$\chi \in [a,b] \quad (\chi \ne \xi_k;\ k=\overline{0,p}) \tag{4.19}$$

выполняется (противоречащее (4.8)) неравенство

$$|\,R(\chi;x)\,| > \frac{M_{n+1}}{\underset{\sim}{\theta}_{n+1}} |\,R(\chi,G_{n+1})\,|, \tag{4.20}$$

и введём "функцию с параметром"

$$f_{\alpha}(t) = \alpha R(t;x) - R(t;\,G_{n+1}). \tag{4.21}$$

Очевидно, при любом значении α, функция (4.21) *будет аннулироваться (кратным образом) в каждом узле интерполяции* задачи (4.3)-(4.7), и тем самым, учитывая (4.15)-(4.16), для числа *изолированных нуль-точек* $Z_T(f_{\alpha};[a,b])$ функции $f_{\alpha}(t)$ на отрезке $[a,b]$, подсчитываемых с учётом их кратностей (в [Б-6] число *изолированных нулей* /с учётом их кратностей/ заданной на отрезке $[a,b]$ гладкой функции $f(t)$ обозначалось через $Z(f;[a,b])$, а здесь к букве Z мы приписываем ещё индекс "Т", поскольку речь идёт только об *изолированных нуль-точках*, без учёта возможно существующих у функции $f(t)$ *изолированных нуль-промежутко)* будет иметь место неравенство

$$Z_T(f_{\alpha};[a,b]) \ge m+1. \tag{4.22}$$

Подберём, далее, в (4.21) параметр $\alpha=\tilde{\alpha}$ так, чтобы соответствующая функция

$$\tilde{f}(t) := \tilde{\alpha} R(t;x) - R(t;\,G_{n+1}) \tag{4.23}$$

имела бы точку χ *дополнительным* своим нулём, что немедленно приводит нас к равенству

$$\tilde{\alpha} = \frac{R(\chi;\,G_{n+1})}{R(\chi;\,x)}, \tag{4.24}$$

реально допустимому в силу предположения (4.20). Из (4.20)-(4.24) тогда имеем, оценку

$$|\,\tilde{\alpha}\,| < \frac{\underset{\sim}{\theta}_{n+1}}{M_{n+1}}. \tag{4.25}$$

Поэтому, учитывая (4.16) и (4.22), для числа *изолированных нуль-точек* $Z_T(\tilde{f};[a,b])$ функции $\tilde{f}(t)$ на отрезке $[a,b]$, подсчитываемых с учётом их кратностей, будет иметь место неравенство

$$Z_T(\tilde{f};[a,b]) \geq (m+1)+1 = [n+N+1]+1 = n+N+2. \quad (4.26)$$

Напомним теперь, что для *обобщённой производной* (*в силу рассматриваемой ECT-системы)* от *произвольной* гладкой функции $x(t) \in C^{(j)}([a,b])$, определяемой формулой

$$D_j[x(t)] =: \frac{W(e_0(t),\ \ldots\ , e_{j-1}(t),\ x(t))}{W(e_0(t),\ \ldots,\ e_{j-1}(t))},$$
$$D_0[x(t)] =: x(t) \quad (j=1,\ 2,\ \ldots), \quad (4.27)$$

справедлива *обобщённая теорема Ролля,* согласно которой *между двумя нулями функции* $x(t)$ *существует по крайней мере один нуль её обобщённой производной* $D_1[x(t)]$, *и, тем самым, между двумя нулями функции* $D_j[x(t)]$ $(j \geq 0)$ *имеется хотя бы один ноль функции* $D_{j+1}[x(t)]$.

(*Примечание 4.* Исторически подобного рода Утверждение, обобщающее классическую теорему французского исследователя Мишеля Ролля /Michel Rolle/ (1652-1719), было высказано, в несколько ином виде, выходцем из Венгрии математиком Дьёрди Пойа /Gyögy Pólya/ (1887-1985). В приведенном виде оно, по-существу, доказывается в [А-12,с. 31-34] и в [Б-3, стр. 97].)

Так вот, согласно этому Утверждению, для $Z_T(D_{n-1}[\tilde{f}];[a,b])$ - числа изолированных *нуль-точек* обобщённой производной *(n -1)-го порядка* от функции $\tilde{f}$ на отрезке $[a,b]$, подсчитываемых с учётом их кратностей – должно выполняться неравенство

$$Z_T(D_{n-1}[\tilde{f}];[a,b]) \geq Z_T(\tilde{f};[a,b]) - (n-1) \geq (n+N+2)-(n-1) = N+3. \quad (4.28)$$

А затем следует пояснить, что для $Z_T(D_{n-1}[\tilde{f}];[a,b])$, при построении $\tilde{f}$ в виде (4.23), такое число изолированных нуль-точек быть не может. Поясним, вкратце, почему.

В самом деле, для альтернансной образующей $G_{n+1}(t)$ имеют место соотношения

$$|D_{n+1}[G_{n+1}(t)]| \geq \underset{\sim}{\theta}_{n+1} \quad (t \in (t_j; t_{j+1});\ j=\overline{0,N}); \quad (4.29')$$

$$\operatorname{sgn} D_{n+1}[G_{n+1}(t)]|_{t\in(t_{j-1},t_j)} = -\operatorname{sgn} D_{n+1}[G_{n+1}(t)]|_{t\in(t_j,t_{j+1})} \quad (j=\overline{1,N}). \quad (4.29'')$$

Отсюда, в интервале

$$(t_j,\ t_{j+1}) \qquad (j=0,1,\ldots,N)$$

для функции

$$D_{n+1}[\tilde{f}(t)] = \tilde{\alpha}\cdot D_{n+1}[x(t)] - D_{n+1}[G_{n+1}(t)], \quad (4.30)$$

учитывая принадлежность $x(t)$ классу $W^{(n+1)}[M_{n+1};\ a,b]$ и наличие для $\tilde{\alpha}$ оценки (4.25), справедливы альтернирующие равенства

$$\operatorname{sgn} D_{n+1}[\tilde{f}(t)]|_{t\in(t_j,\,t_{j+1})} = (-1)^{j+1} \quad (j=0,1,\ldots,N). \quad (4.31)$$

Но (см., например, [А-30, стр. 34]) обобщённые и “обычные” производные связаны между собой рекуррентными соотношениями вида

$$D_j[x(t)] = \frac{d}{dt}D_{j-1}[x(t)] - \frac{\frac{d}{dt}D_{j-1}[e_{j-1}(t)]}{D_{j-1}[e_{j-1}(t)]}D_{j-1}[x(t)] \quad (j \geq 1), \quad (4.32)$$

и потому, при $t \in (t_j, t_{j+1})$ $(j=0,1,\ldots,N)$, должны выполняться равенства

$$D_{n+1}[\tilde{f}(t)] = D_n[e_n(t)]\cdot\frac{d}{dt}\left(\frac{D_n[\tilde{f}(t)]}{D_n[e_n(t)]}\right), \quad (4.33')$$

$$D_n[\tilde{f}(t)] = D_{n-1}[e_{n-1}(t)]\cdot\frac{d}{dt}\left(\frac{D_{n-1}[\tilde{f}(t)]}{D_{n-1}[e_{n-1}(t)]}\right). \quad (4.33'')$$

Поскольку $D_n[e_n(t)]$ на $[a,b]$ нигде не обращается в нуль (и, тем самым, $D_n[e_n(t)]$ на $[a,b]$ сохраняет свой знак), причём эта величина, условно говоря, “стоит как в числителе, так и в знаменателе выражения (4.33’), то из соотношения (4.33’) можно сделать вывод, что знак производной $\frac{d}{dt}(D_n[\tilde{f}(t)])$ определяется знаком функции $D_{n+1}[\tilde{f}(t)]$. Поэтому, учитывая (4.31), имеем

$$\operatorname{sgn}\frac{d}{dt}(D_n[\tilde{f}(t)])|_{t\in(t_j,t_{j+1})} = (-1)^{j+1} \quad (j=0,1,...,N). \tag{4.34}$$

Следовательно, на (t_0,t_1) функция $D_n[\tilde{f}(t)]$ монотонно убывает, на (t_1,t_2) - монотонно возрастает, на (t_2,t_3) - снова монотонно убывает, на (t_3,t_4) - снова монотонно возрастает и т.д.

Поэтому либо на каждом из интервалов (t_j,t_{j+1}) $(j=0,1,...,N)$ функция $D_n[\tilde{f}(t)]$ может иметь не более одного "нечётного" нуля, либо, в крайнем случае, для двух соседних интервалов (t_{j-1},t_j), (t_j,t_{j+1}) не более одного "чётного" нуля в точке t_j. .

(*Примечание 5.* Напомним, что в теории чебышёвских обобщенных полиномов *корень* называют *чётным*, если при переходе через него полином *знака не меняет*, и, в свою очередь, называют его *нечётным*, если либо он совпадает с одним из концов исходного отрезка [*a, b*], либо находится строго внутри [*a, b*], но при переходе через него полином *меняет свой знак* (см., например, [А-12, с. 40])).

Таким образом, для величины $Z_T(D_n[\tilde{f}];[a,b])$ - числа изолированных *нуль-точек* функции $D_n[\tilde{f}(t)]$ на $[a,b]$, подсчитываемых с учётом их кратностей – заведомо будет выполняться неравенство

$$Z_T(D_n[\tilde{f}];[a,b]) \le N+1. \tag{4.35}$$

Аналогично, поскольку $D_{n-1}[e_{n-1}(t)]$ на $[a,b]$ также нигде не обращается в нуль (и, тем самым, $D_{n-1}[e_{n-1}(t)]$ на $[a,b]$ сохраняет свой знак), причём эта величина опять же, условно говоря, "стоит как в числителе, так и в знаменателе выражения (4.33")), то из соотношения (4.33") следует, что знак производной $\frac{d}{dt}(D_{n-1}[\tilde{f}(t)])$ определяется знаком функции $D_n[\tilde{f}(t)]$.

Поясним, что, при этом, для величины $Z_T(D_{n-1}[\tilde{f}];[a,b])$ - числа изолированных *нуль-точек* функции $D_{n-1}[\tilde{f}(t)]$ на $[a,b]$, подсчитываемых с учётом их кратностей – должна выполняться оценка

$$Z_T(D_{n-1}[\tilde{f}];[a,b]) \le N+2. \tag{4.36}$$

Для доказательства оценки (4.36) установим, сначала, по индукции справедливость неравенств

$$Z_T(D_{n-1}[\tilde{f}];[a,t_j]) \le j+1 \quad (j=1,2,...,N+1), \tag{4.37}$$

причём покажем, что если такое соотношение *реализуется как равенство*, то тогда обязательно будет выполняться неравенство

$$(-1)^j D_{n-1}[\tilde{f}(t_j)] \ge 0. \tag{4.38}$$

При $j=1$ утверждение (4.37) – (4.38) объясняется просто. В самом деле, на основании (4.34) функция $D_n[\tilde{f}(t)]$, а значит и $\frac{d}{dt}(D_{n-1}[\tilde{f}(t)])$, внутри промежутка $[a,t_1]$ может иметь тогда самое большее одну изолированную нечётную нуль-точку, причём в этой точке (для графика функции $D_n[\tilde{f}(t)]$) должно быть «монотонно убывающее пересечение "сверху-вниз"» оси *0t*. Отсюда для $D_{n-1}[\tilde{f}(t)]$ указанная точка может быть её единственной точкой внутреннего (локального) максимума, слева от которой эта функция монотонно возрастала бы, а справа – монотонно убывала бы. Тем самым, в такой ситуации у функции $D_{n-1}[\tilde{f}(t)]$ на $[a,t_1]$ самое большее может быть либо два однократных корня (когда значение этой функции в указанной точке максимума строго положительное), либо один двукратный корень (когда значение этой функции в такой точке максимума равно нулю), причём, в обоих ситуациях "зануления" функции $D_{n-1}[\tilde{f}(t)]$), значения $D_{n-1}[\tilde{f}(t_0)]$ и $D_{n-1}[\tilde{f}(t_1)]$ не могут быть положительными. А это и означает, что при $j=1$ имеет место оценка (4.37), где равенство реализуется лишь при соотношении (4.38).

Пусть, теперь, утверждение (4.37) – (4.38) считается справедливым для $1 \le j \le k$ $(k \le N)$, где, для определённости, ***положим k нечётным*** (ситуация чётности k рассматривается аналогичным образом). Выделим, далее, два случая:

1 –ый случай, когда имеет место строгое неравенство

$$Z_T(D_{n-1}[\tilde{f}];[a,t_k]) < k+1, \tag{4.39}$$

и ***2-ой случай***, когда имеет место равенство

$$Z_T(D_{n-1}[\tilde{f}];[a,t_k]) = k+1. \tag{4.40}$$

Разберём по отдельности каждый из этих случаев.

Пусть реализуется **1-ый случай**, **и** пусть, при этом, **справедливо строгое неравенство**

$$D_{n-1}[\tilde{f}(t_k)] > 0. \qquad (4.41)$$

Тогда, на основании монотонного возрастания в интервале (t_k, t_{k+1}) производной $\frac{d}{dt}(D_{n-1}[\tilde{f}(t)])$ **(**в силу монотонного возрастания в этом интервале определяющей её поведение функции $D_n[\tilde{f}(t)]$**)** и непрерывности в точке $t = t_k$ функции $D_{n-1}[\tilde{f}(t)]$, замечаем, что на промежутке $(t_k, t_{k+1}]$ у функции $D_{n-1}[\tilde{f}(t)]$ **либо** вовсе не добавится никакой корень, **либо** добавится лишь один однократный корень **(**с монотонно убывающим пересечением “сверху-вниз” оси *0t* в некоторой внутренней точке указанного промежутка, или же с “аннулированием сверху” на оси *0t* в концевой точке $t = t_{k+1}$ данного промежутка**)**, **либо** добавится один внутренний двукратный корень **(**с касанием “сверху” оси *0t* в некоторой внутренней точке указанного промежутка, как точке своего локального минимума**)**, **либо**, наконец, добавятся два однократных корня **(или** с первоначальным монотонно убывающим пересечением “сверху-вниз” оси *0t* в некоторой внутренней точке указанного промежутка, а затем, с “проходом” через точку своего локального минимума и последующим монотонно возрастающим пересечением “снизу-вверх” в другой внутренней точке указанного промежутка, **или же** “с аннулированием снизу” на оси *0t* в концевой точке $t = t_{k+1}$ данного промежутка**)**. Отсюда в любой из этих ситуаций больше 2-ух корней (с учётом их кратностей) у функции $D_{n-1}[\tilde{f}(t)]$ добавиться не может. Поэтому, при наличии строгого неравенства (4.41), в 1-ом случае мы приходим к оценке

$$Z_T(D_{n-1}[\tilde{f}];[a, t_{k+1}]) \le k+2, \qquad (4.42)$$

причём достижение в ней равенства возможно лишь при условии, когда

$$D_{n-1}[\tilde{f}(t_{k+1})] \ge 0. \qquad (4.43)$$

Если же реализуется **1-ый случай**, но, при этом, **имеет место противоположное (4.41) неравенство**

$$D_{n-1}[\tilde{f}(t_k)] \le 0, \qquad (4.44)$$

то, опять же, на основании указанного поведения производной $\frac{d}{dt}(D_{n-1}[\tilde{f}(t)])$ в интервале (t_k, t_{k+1}) и непрерывности в точке $t = t_k$ функции $D_{n-1}[\tilde{f}(t)]$, замечаем, что на промежутке $(t_k, t_{k+1}]$ у функции $D_{n-1}[\tilde{f}(t)]$ может добавиться, самое большее, лишь один однократный корень (**либо** сразу после точки t_k с монотонно возрастающим пересечением “снизу-вверх” оси *0t* в некоторой внутренней точке промежутка $(t_k, t_{k+1}]$, или же с “аннулированием снизу” на оси *0t* в концевой точке $t = t_{k+1}$ данного промежутка, **либо** с продолженным монотонным убыванием от точки t_k до некоторой внутренней точке своего локального минимума и, лишь “пройдя” её, с последующим монотонно возрастающим пересечением “снизу-вверх” оси *0t* в некоторой другой внутренней точке промежутка $(t_k, t_{k+1}]$, **либо** с “аннулированием снизу” на оси *0t* в концевой точке $t = t_{k+1}$ рассматриваемого промежутка). Поэтому, при наличии в 1-ом случае неравенства (4.44), для величины $Z_T(D_{n-1}[\tilde{f}];[a, t_{k+1}])$ мы приходим к ещё более “жёсткой” (чем (4.42)) её оценке – к неравенству

$$Z_T(D_{n-1}[\tilde{f}];[a, t_{k+1}]) \le k+1. \qquad (4.45)$$

Итак, в **1-ом случае,** когда имеет место строгое неравенство (4.39), *всегда справедлива оценка* (4.42), причём, достижение в ней равенства возможно лишь при условии (4.43). А тогда, вспоминая о нечётности k, “приходим” к неравенству

$$(-1)^{k+1} D_n[\tilde{f}(t_{k+1})] \ge 0. \qquad (4.46)$$

Тем самым в 1-ом случае утверждение (4.37)-(4.38) “индуктивно” обосновано.

Рассмотрим теперь **2-ой случай**, когда имеет место равенство (4.40). По индуктивному предположению это возможно лишь при условии

$$D_{n-1}[\tilde{f}(t_k)] \le 0, \qquad (4.47)$$

поскольку k нечётно. Следовательно, рассуждая также, как и в 1-ом случае при наличии неравенства (4.44), замечаем, что на промежутке $(t_k, t_{k+1}]$ у функции $D_{n-1}[\tilde{f}(t)]$ может добавиться *самое большее только один однократный корень*. Тем самым, здесь будет иметь место оценка

$$Z_T(D_{n-1}[\tilde{f}];[a, t_{k+1}]) \le k+2, \qquad (4.48)$$

причём равенство в ней возможно лишь при условии $D_{n-1}[\tilde{f}(t_{k+1})] \le 0,$ или же, с учётом нечётности k, при условии

$$(-1)^{k+1} D_n[\tilde{f}(t_{k+1})] \ge 0. \qquad (4.49)$$

А этим утверждение (4.37)-(4.38) “индуктивно” обосновывается и для 2-го случая.

Полагая в утверждении (4.37)-(4.38) $j = N+1,$ мы из него сразу же получаем оценку (4.36), являвшейся целью наших рассуждений.

Итак, с одной стороны для величины $Z_T(D_{n-1}[\tilde{f}];[a,b])$ должно быть справедливо неравенство (4.28), а с другой стороны эта величина должна удовлетворять оценке (4.36). Возникшим противоречием и объясняется, *что точки (4.19), где выполняется неравенство (4.20), быть не может*. Отсюда имеет место оценка (4.8), где в качестве функции $F_{n+1}(t)$ следует взять функцию вида

$$R(t;G_{n+1}) = G_{n+1}(t) - S_n(t;G_{n+1}) = [K_{n+1}(t;a) + 2\sum_{j=1}^{N}(-1)^j \cdot \theta_{n+1}^j \cdot K_{n+1}^+(t;t_j)] -$$

$$[\sum_{i=0}^{n} c_{i,0}(G_{n+1}) \cdot K_i(t;a) + \sum_{j=1}^{N} c_{n,j}(G_{n+1}) \cdot K_n^+(t;t_j)] = [K_{n+1}(t;a) - \sum_{i=0}^{n} c_{i,0}(G_{n+1}) \cdot K_i(t;a)] + \quad (4.50)$$

$$[\sum_{j=1}^{N} \{2 \cdot (-1)^j \cdot \theta_{n+1}^j \cdot K_{n+1}^+(t;t_j) - c_{n,j}(G_{n+1}) \cdot K_n^+(t;t_j)\}].$$

Полагая далее

$$\alpha_i = -c_{i,0}(G_{n+1}) \quad (i = 0,1,...,n),$$
$$\beta_j = -c_{n,j}(G_{n+1}) \quad (j = 1,2,...,n), \quad (4.51)$$

мы и обосновываем высказанное Утверждение.

Замечание 5.

Отметим, что при

$$\nu \geq n \quad (4.52)$$

(то есть, когда сплайн обращается в обобщённый полином из пространства $\Phi_n([a,b])$) в качестве функции $F_{n+1}(t)$ тогда следует взять обобщённый полином из пространства $\Phi_{n+1}([a,b])$ с единичным старшим коэффициентом, удовлетворяющий (согласно (4.11)) “аннулируемостью в узлах (4.4.) с кратностями (4.5)” . Иными совами в этом случае

$$F_{n+1}(t) = V_{n+1}(t), \quad (4.53)$$

где

$$V_{n+1}(t) = \frac{W\begin{pmatrix} e_0, & \dots & , e_n, & e_{n+1} \\ \eta_0, & \dots & , \eta_n, & t \end{pmatrix}}{W\begin{pmatrix} e_0, & \dots & , e_n \\ \eta_0, & \dots & , \eta_n \end{pmatrix}}, \quad (4.54)$$

а

$$\{\eta_0 \leq \eta_1 \leq ... \leq \eta_n\} \in [a,b] \quad (4.55)$$

- перенумерованные “сплошной нумерацией” набор интерполяционных узлов (4.4), повторяемых согласно их кратностям (4.5) (подробнее об использовании чебышёвского обобщённого полинома из пространства $\Phi_{n+1}([a,b])$ вида (4.54)-(4.55) для оценки погрешности интерполяции обобщённым полиномом из пространства $\Phi_n([a,b])$ см. [А-13, с. 78-93], а также [Б-5]). В этом смысле доказанная теорема служит “обобщением” соответствующих результатов статьи [Б-5].

On the theory of interpolation in the space of smooth Tchebycheff generalized-polynomial splines

V.B.Demidovich

Abstract: In the space of smooth Tchebycheff generalized-polynomial splines are introduced special splines for the effective representation of the respective interpolation splines and for the estimates of their errors

Keywords: smooth Tchebycheff systems, generalized polynomials, generalized-polynomial splines, interpolation, estimate of error.

ЛИТЕРАТУРА

А. КНИГИ

[А-1] С.Карлин, В.Стадден «Чебышёвские системы и их применение в анализе и статистике». Москва, "Наука", 1976.
[А-2] М.Г.Крейн, А.А.Нудельман «Проблемы моментов Маркова и экстремальные задачи». Москва, "Наука", 1973.
[А-3] Л.Коллатц, В.Крабс «Теория приближений: чебышёвские приближения и их приложения». Москва, "Наука", 1978.
[А-4] Н.П.Корнейчук «Точные константы в теории приближений». Москва, "Наука", 1987.
[А-5] Н.П.Корнейчук «Сплайны в теории приближения». Москва, "Наука", 1984.
[А-6] С.Б.Стечкин, Ю.Н.Субботин «Сплайны в вычислительной математике». Москва, "Наука", 1976
[А-7] Ю.С.Завьялов, Б.И.Квасов, В.Л.Мирошниченко «Методы сплайн-функций». Москва, "Наука", 1980.
[А-8] Г.Полиа, Г.Сеге «Задачи и теоремы из анализа: I-II». Москва, "Наука", 1978.
[А-9] В.Н.Малозёмов, А.Б.Певный «Полиномиальные сплайны». Ленинград, "Изд-во ЛГУ", 1986.
[А-10] G.Nürnberger «Approximation by Spline Functions». Berlin, "Springer", 1989.
[А-11] В.М.Тихомиров «Некоторые вопросы теории приближений». Москва, "Изд -во Моск. ун -та", 1976.
[А-12] В.Б.Демидович «Приближённые вычисления с помощью обобщённых полиномов из чебышёвских пространств: чебышёвские обобщённые полиномы». Москва, "Изд -во Моск. ун -та", 1990.
[А-13] В.Б.Демидович «Приближённые вычисления с помощью обобщённых полиномов из чебышёвских пространств: простое интерполирование, кратное интерполирование, формулы тейлоровского типа». Москва, "Изд -во Моск. ун -та", 1994.

Б. СТАТЬИ

[Б-1] В.С.Романов «Кратная интерполяция чебышёвскими сплайнами».
Методы вычислений (Ленинград), т. 13 (1983), 186-202.
[Б-2] В.Б.Демидович, Г.Г.Магарил-Ильяев, В.М.Тихомиров «Об экстремумах линейных функционалов на конечномерных пространствах». Успехи математических наук, т. 55 (2000), № 4, 133-134 (переведена на английский: *V.B.Demidovich, G.G.Magaril-Ilyaev, V.M.Tikhomirov «Extrema of linear functionals on finite-dimensional spaces». USA, «AMS, Providence, R.I., "Transl. of Math. Journals: Communications of the Moscow Mathematical Society», (2001), 1143-1144*).
[Б-3] В.Б.Демидович, Г.Г.Магарил-Ильяев, В.М.Тихомиров «Экстремальные задачи для линейных функциионалов на чебышёвских пространствах». Фундаментальная и прикладная математика, т. 11 (2005), № 2, 87-100 (переведена на английский: *V.B.Demidovich, G.G.Magaril-Ilyaev, and V.M.Tikhomirov «Extremal problems for linear functional on the Tchebycheff spaces».Springer International Publishing, Journal of Mathematical Sciences, v. 142 (2007), № 2, 1923-1932*).
[Б-4] В.Б.Демидович, А.С.Кочуров «О некоторых экстремальных задачах в конечно-мерных чебышёвских пространствах». Труды НИИСИ РАН, т. 4 (2014), № 2, 132-141.
[Б-5] В.Б.Демидович «Об интерполяции функций обобщёнными полиномами, построенными на базе гладких чебышёвских систем». Труды НИИСИ РАН, т. 6 (2016), № 1, 102-114.
[Б-6] В.Б.Демидович «Об интерполяции функций обобщённо-полиномиальными сплайнами, построенными на базе гладких чебышёвских систем». Труды НИИСИ РАН, т. 6 (2016), № 2, 129-142.

О специальных решениях уравнений в подгруппах конечного поля

Ю.Н. Штейников

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: yuriisht@rambler.ru

Аннотация: Излагается подход к изучению величины, связанной с числом решений специальных уравнений в конечных множествах и некоторых обобщений.

Ключевые слова: плотность множества, делимость, конечное поле.

Пусть $A \subset \mathsf{N}$ - множество, замкнутое относительно операции умно-жения, то есть если $a_1, a_2 \in A$, то $a_1 a_2 \in A$. Такие множества называют полугруппами. В частности, в качестве A можно взять множество

$$A = \{n \in \mathsf{N} : n \in G(\bmod m)\},$$

где $m \in N$, а G - мультипликативная подгруппа группы Z_m^*, то есть множества обратимых элементов вычетов по модулю m.

При изучении числа решений линейных уравнений над конечным полем в подгруппах полезно иметь оценки на плотность этих множеств в специальных интервалах. О последней величине и пойдет речь в этой статье.

Нас будет интересовать случай, когда для некоторого натурального q и $0 < \nu < 1$ справедливо неравенство:

$$|\{n \in A; n \le q\}| < q^{\nu}. \qquad (*)$$

Пусть $x>0$ и пусть

$$f(x) := |A \cap [1,x]|.$$

В работе [1] получены оценки на количество чисел, не превосходящих n, но принадлежащих подгруппе порядка t группы Z_p^*. Эти оценки содержательны, когда t мало по сравнению с p. Из наших же результатов вытекают оценки в случае, когда t растет как степень p, а n мало.

В работе [2] были уже получены нетривиальные верхние оценки на $f(n)$, когда n растет медленнее, чем произвольная степень q. Целью данной статьи является указать идеи для получения более сильного варианта теоремы, чем в [2].

Приведем утверждение, доказанное в работе [2].

Теорема 1. *Пусть A -- множество, замкнутое относительно операции умножения, удовлетворяющее условию (*) для некоторого фиксированного $\nu \in (0,1)$, и пусть задано x. Если*

$$\gamma := \frac{\operatorname{loglog} x}{\operatorname{loglog} q}, \quad \log x = o(\log q),$$

то

$$f(x) \le x^{1-\max\{L_\gamma, M_\gamma\}+o(1)}, \quad q \to \infty,$$

где

$$L_\gamma = \gamma\left(\frac{1-\nu}{1-\gamma+\sqrt{(1-\gamma)^2+\gamma(1-\nu)}}\right)^2,$$

а

$$M_\gamma = \frac{(1-\nu)^2\gamma}{4(1-\gamma)} \text{ при } \gamma \le \frac{2}{3-\nu},$$

и

$$M_\gamma = 2-\nu-\frac{1}{\gamma} \quad \text{при } \gamma > \frac{2}{3-\nu}.$$

Основной результат настоящей главы - доказать следующий более сильный вариант Теоремы 1:

Теорема 2. *Пусть A - множество, замкнутое относительно операции умножения, удовлетворяющее условию (*) для некоторого фиксированного $\nu \in (0,1)$ и пусть задано x. Если*

$$\gamma := \frac{\operatorname{loglog} x}{\operatorname{loglog} q}, \quad \log x = o(\log q),$$

то

$$f(x) \le x^{1-C_\gamma+o(1)}, \quad q \to \infty,$$

где

$$C_\gamma = \gamma(\frac{\gamma - 1 + \sqrt{\gamma^2 + \nu - 2\gamma\nu}}{2\gamma - 1})^2 \quad для \quad \gamma \neq \frac{1}{2}$$

и

$$C_{\frac{1}{2}} := \lim_{\gamma \to \frac{1}{2}} C_\gamma = \frac{1}{2}(1-\nu)^2.$$

Схема доказательства теоремы 2 совпадает со схемой доказательства теоремы 1. Но при этом в доказательстве будем пользоваться одной более точной оценкой, которая и приводит к результату теоремы 2.

Дополнительные сведения.

Мы будем использовать определения и обозначения из работы [2]. Кроме того, нам потребуются оценки для множеств чисел, у которых все простые делители малы.

Для натурального n пусть $P^+(n)$ обозначает наибольший простой делитель числа n, $P^+(1) = 1$.

Для $x \geq y \geq 2$ полагаем:

$$\psi(x, y) = |\{n \leq x : P^+(n) \leq y\}|.$$

Известны оценки на количество гладких чисел в заданных интервалах.

Предположим, что задано целое y. Каждое натуральное n представим в виде произведения $n = n_1 n_2$, так что если простое p делит n_1, то $p \leq y$, а если делит n_2, то $p > y$. Пусть также даны x, z. Определим множество:

$$N(x, y, z) = \{n \leq x : n_1 > z\}.$$

Идея доказательства Теоремы 2.

Разбиваем дизъюнктно множество A из интервала [1,x] на два множества C и D. При этом положим, что элементы множества C представляют собой элементы из A, у которых гладкая часть больше z, а во множество D входят все остальные элементы из A.

Размер множества C оцениваем с помощью величины

$$N(x, y, z) = \{n \leq x : n_1 > z\}.$$

Элементы множества D обладают тем свойством, что произведение DD растет по сравнению с размером D. Этот факт и позволяет получить верхнюю оценку для размера множества D.

Публикация выполнена в рамках Государственного задания на выполнение фундаментальных научных исследований по программам РАН по теме (проекту) «1.5П Исследование и разработка высокоэффективных алгоритмов в компьютерной алгебре, алгебраической геометрии и теории чисел» (0065-2015-0122).

On special equations over subgroups in the finite fields

Yu. N. Shteinikov

Abstract: An approach is presented to the study of the density of sets with good multiplicative properties.

Keywords: density sets, divisibility,finite field

Литература

1. J.Bourgain, S.Konyagin, I.Shparlinski. Distribution of elements of cosets of small subgroups and applications // International Math Research Notices. 2012, v. 201, №9, p. 1968--2009
2. Ю.Н.Штейников. О распределении элементов полугрупп натуральных чисел // Чебышевский сборник. 2012. т. 13, № 3, с. 91–99.
3. Ю.Н.Штейников. О распределении элементов полугрупп натуральных чисел II // Чебышевский сб., 2016, том 17, выпуск 3, страницы 197–203.

О плотности полугрупп натуральных чисел

Ю.В. Кузнецов[1], Ю.Н. Штейников[2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1]yukuz61@rambler.ru, [2]yuriisht@gmail.com

Аннотация: Изучается плотность полугрупп на коротких интервалах.

Ключевые слова: плотность множества, делимость.

Введение

Пусть $A \subset N$ - множество, замкнутое относительно операции умножения, то есть если $a_1, a_2 \in A$, то $a_1 a_2 \in A$. Такие множества называют полугруппами.

В частности, можно взять множество

$$A = \{n \in N : n \in G (\mathrm{mod}\, m)\},$$

где $m \in N$, а G-мультипликативная подгруппа группы Z_m. Мы будем изучать такие множества среди натуральных чисел.

Нас будет интересовать случай, когда для некоторого натурального q и $0 < \nu < 1$ справедливо неравенство:

$$|\{n \in A;\ n \le q\}| < q^{\nu}. \qquad (*)$$

Пусть для $x>0$ определена функция

$$f(x) = |A \cap [1,x]|.$$

Основная цель настоящей статьи – доказать следующую теорему:

Теорема 1. *Пусть A - множество, замкнутое относительно операции умножения, удовлетворяющее условию (*) для некоторого фиксированного $\nu \in (0,1)$ и пусть задано x. При*

$$\gamma := \frac{\log\log x}{\log\log q} \quad и \quad \log x = o(\log q),$$

тогда имеем

$$f(x) \le x^{1 - C_\gamma + o(1)},\ q \to \infty,$$

где

$$C_\gamma = \gamma\left(\frac{\gamma - 1 + \sqrt{\gamma^2 + \nu - 2\gamma\nu}}{2\gamma - 1}\right)^2 \quad если \quad \gamma \ne \frac{1}{2}$$

и

$$C_{\frac{1}{2}} := \lim_{\gamma \to \frac{1}{2}} C_\gamma = \frac{1}{2}(1-\nu)^2.$$

Вспомогательные утверждения

Нам потребуются оценки для множеств чисел, у которых все простые делители малы.

Для натурального n пусть $P^+(n)$ обозначает наибольший простой делитель числа n, $P^+(1)=1$.

Для $x \ge y \ge 2$ полагаем:

$$\psi(x,y) = |\{n \le x : P^+(n) \le y\}|.$$

Известна следующая теорема (см. [3]):

Теорема 2. *Пусть $x \ge y \ge 2$,* $v = \frac{\log x}{\log y}$. *Тогда для любого $\varepsilon>0$ на множестве $v \le y^{1-\varepsilon}$ имеет место неравенство:*

$$\psi(x,y) = x v^{-v(1+o(1))},$$

при $v \to \infty$.

Предположим теперь, что задано целое y. Каждое натуральное n представим в виде произведения $n = n_1 n_2$, так что если простое p делит n_1, то $p \le y$, а если делит n_2, то $p > y$. Пусть, также, даны *два различных числа* x и z. Определим множество:

$$N(x,y,z) = \{n \le x : n_1 > z\}.$$

Мы хотим оценить сверху количество элементов множества $N(x,y,z)$.

На довольно большой области изменения x, y, z была получена асимптотика $N(x,y,z)$ в работе [4]. Однако в работе [1] был получен соответствующий результат при более слабых ограничениях на параметры x, y, z. Ниже приводится его формулировка:

Лемма 1. *Пусть $\varepsilon>0$ фиксировано, q, x - достаточно большие. Предположим, что положительные вещественные α, β, γ удовлетворяют условиям*

$$\beta < 1, \quad \gamma \le \alpha(1-\varepsilon), \quad 0 \le \frac{\beta}{\alpha} \le 1/\varepsilon,$$

Причём

$$y := (\log q)^{\alpha} \ge 2, \quad z := x^{\beta}.$$

Тогда если $\frac{\log x}{\log\log q} \to \infty$ при $q \to \infty$,

то

$$|N(x,y,z)| \le x^{1 - \frac{\beta\gamma}{\alpha} + o(1)}.$$

Нам также понадобится другое утверждение, доказанное в [1], приводимое ниже:

Лемма 2. *Количество делителей числа* n < Q, *не превосходящих z, не превосходит*

$$\psi\big(z,(1+o(1))\log Q\big) \text{ при } Q \to \infty.$$

Теперь все готово для доказательства Теоремы 1.

Доказательство Теоремы 1.

Пусть $\gamma:=\frac{\log\log x}{\log\log q}$. Введем положительные вещественные параметры $\alpha > 1$ и β, удовлетворяющие условию Леммы 1 для некоторого фиксированного $\varepsilon > 0$.

Положим, что $y = (\log q)^{\alpha}$, $z=x^{\beta}$. Каждое натуральное n представим в виде произведения $n = n_1 n_2$, так что если простое p делит n_1, то $p \le y$, а если p делит n_2, то $p > y$. Разделим элементы множества $A \cap [1,x]$ на два подмножества A' и A'', так что

$$A \cap [1,x] = A' \cup A'',$$

причём, по определению, положим

$$A'':=\{n \in A \cap [1,x]: n_1>z\}.$$

Совершенно ясно, что

$$f(x) = |A'| + |A''|.$$

I. Оценим, сначала, размер множества A''. По Лемме 1 сразу получаем:

$$|A''| \le N(x,y,z) \le x^{1-\frac{\beta\gamma}{\alpha}+o(1)},\ q \to \infty.$$

II. Теперь перейдем к оценке $|A'|$. Для этого рассмотрим множество $B := \{m_1 \dots m_r\}$, где $r = [\frac{\log q}{\log x}] = [(\log q)^{1-\gamma}]$ и $m_1, \dots, m_r \in A'$. Из определения r следует, что если $m \in B$, то $m \le q$. Так как произведение чисел из A' является числом из A, то легко видеть, что $|B| \le |A \cap [1,q]| \le q^{\nu}$.

Теперь оценим снизу $|B|$. Пусть каждый $m_i \in A'$ по аналогии представим в виде $m_i=m_{1,i}m_{2,i}$, так что если простое p делит $m_{1,i}$, то $p \le y$, а если p делит $m_{2,i}$, то $p > y$. Определим N_1, N_2 из равенства:

$$m = m_1 \dots m_r = m_{1,1} \dots m_{1,r} m_{2,1} \dots m_{2,r},$$

где

$N_1=m_{1,1} \dots m_{1,r}$ и $N_2=m_{2,1} \dots m_{2,r}$.

Возьмем произвольный элемент $m \in B$, и оценим сверху число представлений его в виде произведения r множителей, где каждый принадлежит A'.

Пусть $m = N_1 N_2$. Оценим количество представлений для N_2 в виде произведения r чисел $m_{2,1} \dots m_{2,r}$: $N_2 = m_{2,1} \dots m_{2,r} = p_1 \dots p_s$, где все $p_i > y$ и являются простыми числами. Видим, что

$$s \le [\frac{\log N_2}{\log y}] \le \frac{\log N_2}{\alpha \log\log q}.$$

Каждый делитель p_i (i=1, … ,s) может входить в разложение некоторого $m_{2,j}$ (j=1, … ,r). Значит количество представлений числа N_2 не превосходит $r^s \le N_2^{\frac{1-\gamma}{\alpha}}$.

Теперь оценим количество представлений для фиксированного N_1 в виде $N_1 = m_{1,1} \dots m_{1,r}$, когда $m_{1,i} \in A'$. Покажем, что количество таких представлений числа N_1 не превосходит

$$N_1^{1-\gamma} q^{o(1)},\quad q \to \infty.$$

Зафиксируем достаточно большое натуральное число J. Введем по определению интервалы $\Delta_j:=\left[z^{\frac{j-1}{J}}, z^{\frac{j}{J}}\right)$, $j \in [1,J]$.

Каждый из множителей $m_{1,i}$ (i=1, … ,r) попадает в один из интервалов Δ_j (j=1, … ,J). Ясно, что количество способов распределения r чисел $m_{1,i}$ (i=1, … ,r) по интервалам Δ_j не превосходит $J^r=q^{o(1)}$, $q \to \infty$.

Пусть теперь каждый $m_{1,i}$ (i=1, … ,r) относится к какому-нибудь из Δ_j (j=1, … ,J). Если, например, $m_{1,1} \in \Delta_j$, то $m_{1,1} \le z^{\frac{j}{J}}$ и $m_{1,1}$ является делителем числа $N_1 < q$. По Лемме 2 количество таких $m_{1,1}$ не превосходит $\psi\left(z^{\frac{j}{J}}, (1+o(1))\log q\right)$. А пользуясь Теоремой 2, тогда получаем

$$\psi\left(z^{\frac{j}{J}}, (1+o(1))\log q\right) = z^{\frac{j}{J}(1-\gamma+o(1))},\ q \to \infty.$$

Таким образом, если $m_{1,i} \in \Delta_j$, то количество возможностей для числа $m_{1,i}$ не превосходит $z^{\frac{j}{J}(1-\gamma+o(1))}$ и

$$z^{\frac{j}{J}(1-\gamma+o(1))} \le z^{(\frac{j-1}{J}+\frac{1}{J})(1-\gamma+o(1))} \le m_{1,i}^{1-\gamma+o(1)} z^{\frac{1}{J}(1-\gamma+o(1))}.$$

Поэтому, если известно, что каждый из $m_{1,i}$ (i=1, … ,r) принадлежит какому-то из Δ_j, то количество представлений числа N_1 в виде произведения чисел $m_{1,i}$ не превосходит $N_1^{1-\gamma+o(1)} q^{\frac{\beta}{J}(1-\gamma+o(1))}$. Умножив эту величину на число способов распределения чисел $m_{1,i}$ по интервалам Δ_j, которое равно $q^{o(1)}$, получим верхнюю оценку на искомое число представлений N_1 в виде произведения $m_{1,1} \dots m_{1,r}$. Оно не превосходит $N_1^{1-\gamma+o(1)} q^{\frac{\beta}{J}(1-\gamma)+o(1)}$. В силу произвольности J эта величина есть $N_1^{1-\gamma+o(1)} q^{o(1)}$, $q \to \infty$.

Итак, мы оценили количество представлений чисел N_1 и N_2 в виде произведения $m_{1,1} \dots m_{1,r}$ и, соответственно, $m_{2,1} \dots m_{2,r}$. Число же представлений элемента $m \in B$ в виде произведения $m_1 \dots m_r$, где $m_i \in A'$, не превосходит произведения числа представлений для N_1 и N_2, то есть не превосходит величины $N_1^{1-\gamma} N_2^{\frac{1-\gamma}{\alpha}}$.

При $\alpha > 1$ наибольшее значение, которое эта величина может принимать, равно $q^{(1-\gamma)(\beta+\frac{1-\beta}{\alpha})+o(1)}$. Отсюда получается и нижняя оценка для B:

$$\frac{|A'|^r}{q^{(1-\gamma)(\beta+\frac{1-\beta}{\alpha})+o(1)}} \le |B| \le q^{\nu}.$$

Следовательно, для $|A'|$ справедливо:

$$|A'| \le x^{\nu+(1-\gamma)(\beta+\frac{1-\beta}{\alpha})+o(1)}.$$

Вспомним, также, что

$$|A''| \le x^{1-\frac{\beta\gamma}{\alpha}+o(1)}.$$

Складывая эти две оценки, мы получим “заготовленную” оценку для $f(x)$.

Теперь осталось выбрать параметры $\alpha > 1$ и β для оптимизации найденной оценки.

Если $\gamma \neq \frac{1}{2}$, то возьмем следующие параметры:

$\beta := \frac{\gamma-1+\sqrt{\gamma^2-2\gamma\nu+\nu}}{2\gamma-1}$, $\alpha := \frac{1}{\beta}$.

Если же $\gamma = \frac{1}{2}$, то положим $\beta = \frac{1}{\alpha} = 1 - \nu$.

Подставляя такие параметры, мы и завершим доказательство Теоремы 1.

Заключение

Теперь получим нижние оценки на величину $f(x)$, когда x = exp{(log q)$^{\gamma}$}.

Возьмём любое число $0 < \nu < 1$ и положим, что A_q – полугруппа y-гладких чисел, где

$$y = (\log q)^{\lambda},$$

причём $\varepsilon = \frac{1}{1-\nu+u}$, а u - малое число.

Пользуясь Теоремой 2 о количестве гладких чисел при достаточно больших q, получаем, что

$$|A_q[1,q]| < q^{\nu}.$$

Тогда выполнено неравенство (*).

Пользуясь же Теоремой 1, получаем, что

$$|A_q[1,e^{(\log q)^{\gamma}}]| = x^{1-\gamma+\gamma\nu-\varepsilon'\gamma+o(1)}, \quad q \to \infty,$$

где $\varepsilon' > 0$ - некоторое малое число. Причём, верхняя оценка для величины C_{γ}, с точностью до слагаемых малых порядков, тогда не превосходит $\gamma - \gamma\nu$.

Заметим, что оценка, полученная Теоремой 2 для величины $C_{\gamma,}$ хуже предыдущей. Возможно, именно она близка к *оптимальной*.

Публикация выполнена в рамках Государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) «Исследования по теоретико-числовым, геометрическим и комбинаторным свойствам алгебраических групп и их приложениям». (0065-2014-0002).

On a density of semigroup of positive integers.

Yu. N. Shteinikov

Abstract: We study the distribution of semigroups of integers on small intervals.

Keywords: density sets, divisibility, finite field

Литература

1. Ю.Н.Штейников. О распределении элементов полугрупп натуральных чисел // Чебышевский сборник. 2012, т. 13, № 3 .с. 91–99.
2. Ю.Н.Штейников. О распределении элементов полугрупп натуральных чисел II // Чебышевский сб., 2016, т. 17, № 3, с. 197–203.
3. A. Hildebrand, G.Tenenbaum. Integers without large prime factors // J. Theorie des Nombres de Bordeaux, 1993. v. 5, № 2. p. 411-484.
4. W.Banks, I.Shparlinski. Integers with a large smooth divisor // Electronic journal of combinatorial number theory, 2007, v. 7.