Федеральное государственное бюджетное учреждение

Национальный исследовательский центр «Курчатовский институт»

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

# ТРУДЫ НИИСИ РАН

ТОМ 14 № 1

## МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

## ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2024

**Тематика номера:**
Моделирование физических процессов в микро- и наноэлектронике, проблемы генерации псевдослучайных чисел, вопросы программирования, моделирование когнитивных процессов

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

**The topic of the issue:**

Modeling of physical processes in micro- and nanoelectronics, pseudorandom number generation problems, programming issues, modeling of cognitive processes

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

Заведующий редакцией: В.Е.Текунов

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

---

## СОДЕРЖАНИЕ



## CONTENT

# Проблемы и направления развития микросхем с перевёрнутым кристаллом

**А.М. Баранов[1], А.А. Подковыров[2], А.В. Андреев[3]**

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

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

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, alandreev@cs.niisi.ras.ru

**Аннотация.** В работе описываются проблемы и направления развития микросхем с перевёрнутым кристаллом. Рассматриваются особенности и свойства полимерных и LTCC подложек. Приводятся основные характеристики микросхем, разработанных в ФГУ ФНЦ НИИСИ РАН. Описана технология «система в корпусе» и ее преимущества.

**Ключевые слова:** технология перевернутого кристалла, низкотемпературные керамические подложки, полимерные подложки, система в корпусе

## 1. Введение

Мировой рынок полупроводниковых устройств постоянно стремится к созданию более компактных, быстрых, энергоэффективных микросхем для различных применений, начиная от мобильных устройств до серверов. Технология микросхем с перевёрнутым кристаллом является основным направлением создания высокопроизводительных микросхем. Однако при переходе к этой технологии возникают вызовы, связанные с производством, надежностью, применяемыми материалами, способностью противостоять внешним воздействиям. Исследование и разработка различных подложек, новых материалов, методов сборки и технологий тестирования играют важную роль в преодолении этих проблем и в успешном внедрении микросхем с перевёрнутым кристаллом на рынок.

## 2. Особенности полимерных и низкотемпературных керамических подложек

Подложка является одной из ключевых составляющих микросхемы, предоставляющей механическую поддержку и электрическую изоляцию для компонентов. Перевёрнутые кристаллы обычно изготавливаются по технологическим нормам от 90 нм и ниже и могут содержать матрицу из более 10 тысяч контактов (бампов) [1]. Соответствующие подложки должны обеспечить такой кристалл надлежащим количеством входных и выходных сигналов, а также питанием и землёй. Это достигается с помощью матрицы, как правило, шариковых выводов (BGA) необходимого размера. В ряде случаев шариковые выводы могут не устанавливаться, тогда такая микросхема определяется как безвыводная (например, как LGA – Land Grid Array).

В зависимости от конкретных требований и характеристик микросхемы, используются полимерные или низкотемпературные керамические подложки. Каждая из них имеют свои технологические особенности.

### 2.1. Полимерные подложки

Материал полимерных подложек, такой как Bismaleimide Triazine (BT resin) и его аналоги обладает рядом уникальных свойств (термостойкость и высокие температуры стеклования (>250°C), низкие диэлектрические константы, хорошая адгезия к проводящим материалам и композитам, огнестойкость) [2].

#### 2.1.1. Воздействие повышенных температур

Высокие температуры стеклования полимеров в составе подложки позволяют безболезненно производить кратковременный нагрев микросхемы при её монтаже на печатную плату модуля. Однако длительное воздействие повышенной температуры может приводить к искажению формы подложки. Так, например, после электротермотренировки при 140°C нами наблюдалось отклонение микросхемы от плоской формы в сторону сферичности, не позволяющее припаять микросхему на печатную плату модуля. Поэтому с своей документации мы не рекомендуем эксплуатировать микросхемы на полимерных подложках при температурах, превышающих 85°C. Дополнительно для минимизации искажения плоскостности подложки необходимо следить за одинаковостью заполнения металлом симметричных слоёв подложки.

#### 2.1.2. Влагоустойчивость

Полимерный подложки способны поглощать влагу. Поэтому они должны храниться либо в вакуумной упаковке, либо в шкафах сухого хранения. В основном полимерный подложки относятся к уровню3 чувствительности к влажности по стандарту IPC/JEDEC J-STD-020C. Такие микросхемы в соответствии с данным стандартом после вскрытия упаковки должны быть установлены с модули в течении 168 часов. Кроме того, перед монтажом они должны быть выдержаны 24 часа в печи при температуре 125°C для удаления влаги. В противном случае, при монтаже микросхемы вода перейдёт в пар и может

произойти расслоение подложки (рисунок 1).

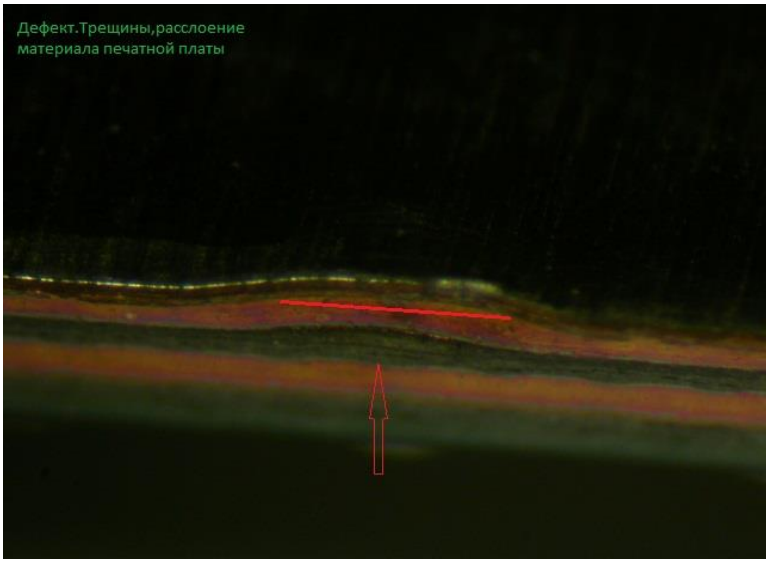



Рис. 1. Расслоения материала печатной платы подложки

### 2.1.3. Число слоёв

Полимерные подложки имеют технологическое ограничение на количество слоёв. Технология для подложек с числом слоёв, превышающим 14 (7-2-7 и более) считается производителем как «рискованная». При это уменьшается выход годных подложек, и, соответственно, сильно возрастает их стоимость.

## 2.2. Подложки из низкотемпературной керамики

Технология низкотемпературной керамики (LTCC, Low Temperature Co-fired Ceramics) в настоящее время быстро развивается и успешно используется для производства ВЧ- и СВЧ-схем низкой и средней степени интеграции, телерадиовещания, телекоммуникаций, мобильной связи, портативных компьютеров и других приборов. Основными потребителями изделий на основе LTCC являются производители электронной аппаратуры, выпускающие как изделия массового потребления, так и двойного назначения.

При этом LTCC подложки не уступают полимерным подложкам в эффективности передачи высокоскоростных сигналов.

### 2.2.1. Стоимость

Стоимость LTCC подложки для перевёрнутых кристаллов зависит от материала проводника. Стоимость подложки с серебряными проводниками может превышать стоимость подложки с медными проводниками в 10 – 100 раз (в зависимости от размера). Это делает использование серебросодержащих паст в подложках коммерчески не приемлемым для массового производства.

Медносодержащие LTCC подложки несколько дороже полимерных подложек. Однако такие подложки являются безальтернативным решением для высокоскоростных микросхем с перевёрнутым кристаллом в условиях, когда окружающая среда не позволяет использовать полимерные подложки (расширенный диапазон температур, повышенная влажность, длительное воздействие радиации, вакуум).

### 2.2.2. Механическая прочность

Для микросхем с высокоскоростными интерфейсами обмера информацией одним их основных требований является соблюдение необходимого волнового сопротивления для дифференциальных пар. Геометрические параметры проводников определяются моделированием исходя из данных производителя подложек (параметры проводника и изолятора, толщины слоёв керамики и т.п.). Например, в разработанной ФГУ ФНЦ НИИСИ РАН микросхеме маршрутизатора RapidIO для вывода высокоскоростных сигналов RapidIO от перевёрнутого кристалла до шариковых выводов на подложке с обеспечением необходимого волнового сопротивления дифференциальных пар было использовано 17 слоёв LTCC керамики толщиной 75 мкм и 50 мкм с медными проводниками. При этом толщина подложки составила 1,15±0,12 мм при размере 31 х 31 мм. Остеклованная керамика такой толщины оказалась довольно хрупкой (см. рисунок 2) и потребовала повышенной осторожности при проведении испытаний.

Рис. 2. Разрушение подложки из LTCC керамики

## 2.3. Сложность подложек

Изначально подложки для перевёрнутых кристаллов более сложные по сравнению с подложками для микросхем с проволочным соединением кристалл-корпус. Это связано с тем, что требуется развести сигналы, земли и питания с огромного количества бампов на кристалле через подложку к шариковым выводам. Задача решается путём использования большого количества слоёв проводников со слепыми переходными отверстиями. Во множестве слоёв удаётся с нужными требованиями развести скоростные сигналы, обеспечить надлежащее питание. При этом количество слоёв ограничивается возможностями производства и ценой подложки.

На рисунке 3 приведены данные по числу слоёв в подложках, разработанных в ФГУ ФНЦ НИИСИ РАН.

В таблице 1 приведены основные размеры разработанных в ФГУ ФНЦ НИИСИ РАН подложек.

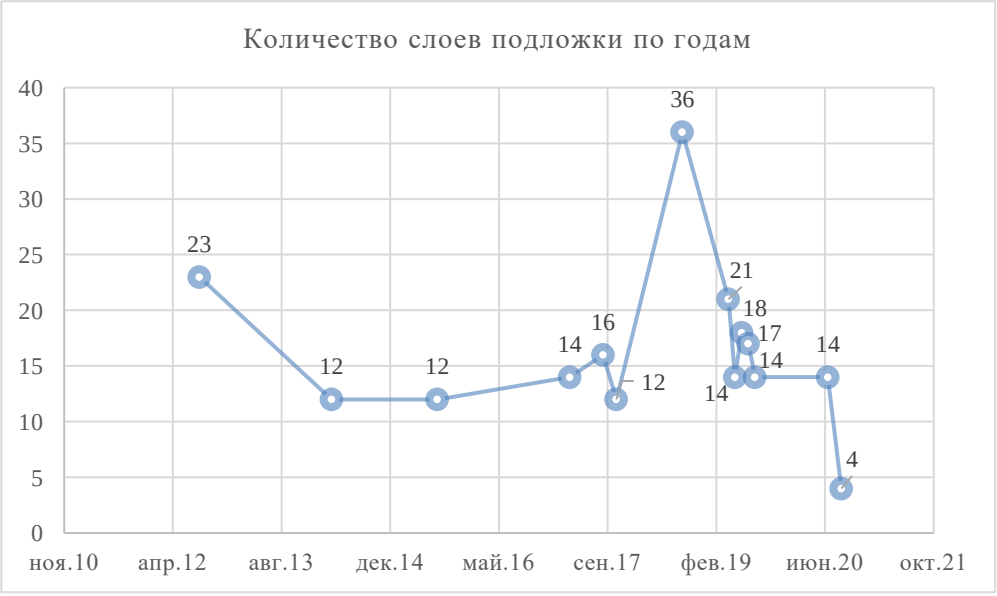



Рис. 3. График количества слоев подложек, разработанных в ФГУ ФНЦ НИИСИ РАН, по годам

Из графика видно, что для полимерных подложек (перечень из таблицы 1), как правило, достаточно 12-14 слоёв подложки чтобы надлежащим образом развести сигналы, земли и питания от кристалла к шариковым выводам. В то время, как для LTCC подложек для решения аналогичной задачи требуется большее количество слоёв: около 20 и выше. Это объясняется тем, что в керамике размеры проводников и зазоров между ними значительно больше, чем в полимерной подложке (~75 мкм против ~15 мкм), и требуется больше слоёв, чтобы развести сигналы.

На графике выделяется высокое количество слоев (36) для LTCC подложки для микросхемы на рисунке 2. Это объясняется наличием в микросхеме большого количества интерфейсов ввода вывода, в том числе 8 каналов последовательного RapidIO (2,5 Гбит/с), для разводки сигналов и питаний которых и потребовалось такое количество слоёв.

Размеры подложек для перевёрнутых кристаллов определяются исходя из двух соображений:

- количество шариковых выводов в матрице должно быть необходимым и достаточным для вывода/вывода всех сигналов и обеспечения микросхемы необходимым питанием;
- по возможности унифицировать матрицы шариковых выводов в минимальное количество типоразмеров.

Из таблицы видно, что до сих пор в основном использовались три размера подложек: 21x21 мм,31x31 мм, 37,5x37,5 мм. Это позволило сэкономить на оснастке для испытания микросхем, в частности использовать унифицированные контактирующие устройства (КУ). На сегодняшний день, в связи с развитием направления по созданию коммерческих микросхем, основным приоритетом становится максимальное уменьшение цены микросхемы при серийном производстве. Возникла необходимость минимизации параметров подложки. Типичном примером такого подхода является промышленный контроллер, разработанный ФГУ ФНЦ НИИСИ РАН. Характеристики подложки для этого контроллера приведены в строке 14 таблицы 1.

Таблица 1. Основные характеристики разработанных flip-chip корпусов

| **Микросхема (порядковый номер)** | **Материал подложки** | **Размер подложки (мм)** | **Матрица шариковых выводов (шаг 1 мм)** | **Кол-во слоев подложки** | **Дата разработки (мес. год)** |
|---|---|---|---|---|---|
| **1** | LTCC | 31 × 31 | 30 x 30 | 23 | авг.12 |
| **2** | полимер | 37.5 × 37.5 | 36 x 36 | 5-2-5 (12) | апр.14 |
| **3** | полимер | 37.5 × 37.5 | 36 x 36 | 5-2-5 (12) | авг.15 |
| **4** | полимер | 31 × 31 | 30 x 30 | 6-2-6 (14) | апр.17 |
| **5** | полимер | 31 × 31 | 30 x 30 | 7-2-7 (16) | сен.17 |
| **6** | полимер | 21 × 21 | 20 x 20 | 5-2-5 (12) | ноя.17 |
| **7** | LTCC | 31 × 31 | 30 x 30 | 36 | сен.18 |
| **8** | LTCC | 31 × 31 | 30 x 30 | 21 | апр.19 |
| **9** | полимер | 37.5 × 37.5 | 36 x 36 | 6-2-6 (14) | май.19 |
| **10** | LTCC | 31 × 31 | 30 x 30 | 18 | июн.19 |
| **11** | LTCC | 31 × 31 | 30 x 30 | 17 | июл.19 |
| **12** | полимер | 31 × 31 | 30 x 30 | 6-2-6 (14) | авг.19 |
| **13** | полимер | 37.5 × 37.5 | 36 x 36 | 6-2-6 (14) | июл.20 |
| **14** | полимер | 17 × 17 | 16 x 16 | 1-1-1-1 (4) | сен.20 |

## 3. Рост сложности кристаллов процессоров и контроллеров

Производство чипов для микросхем является высоко конкурентным бизнесом. Ведущие производители вкладывают огромные деньги в развитие технологии, позволяющей еще уменьшить размеры транзисторов и проводников на кристалле. Развитие технологии описывается законом Мура: количество транзисторов на кристалле удваивается

примерно каждые два года. Закон был сформулирован основателем Intel Гордоном Муром в 1965 году, и это закон продолжает работать вплоть до сегодняшнего дня (рис. 4) [3]. Здесь приведен анализ закона Мура, где сравнивается прогноз с достигнутым уровнем. Для построения графика с использованием математического ряда авторы выбрали две отправные точки: данные Мура (64 компонента в 1965 году и число компонентов на первой ИС Intel 4040:

Прогноз по скорректированному Муром закону с начальной величиной в 64 компонента оказался явно более оптимистичным (синяя линия). А прогнозный ряд с отправной от Intel 4040 величины (красная линия) практически совпадает с реально достигнутыми результатами (чёрная линия). По оси X указаны годы, а по оси Y указано число транзисторов на кристалле.

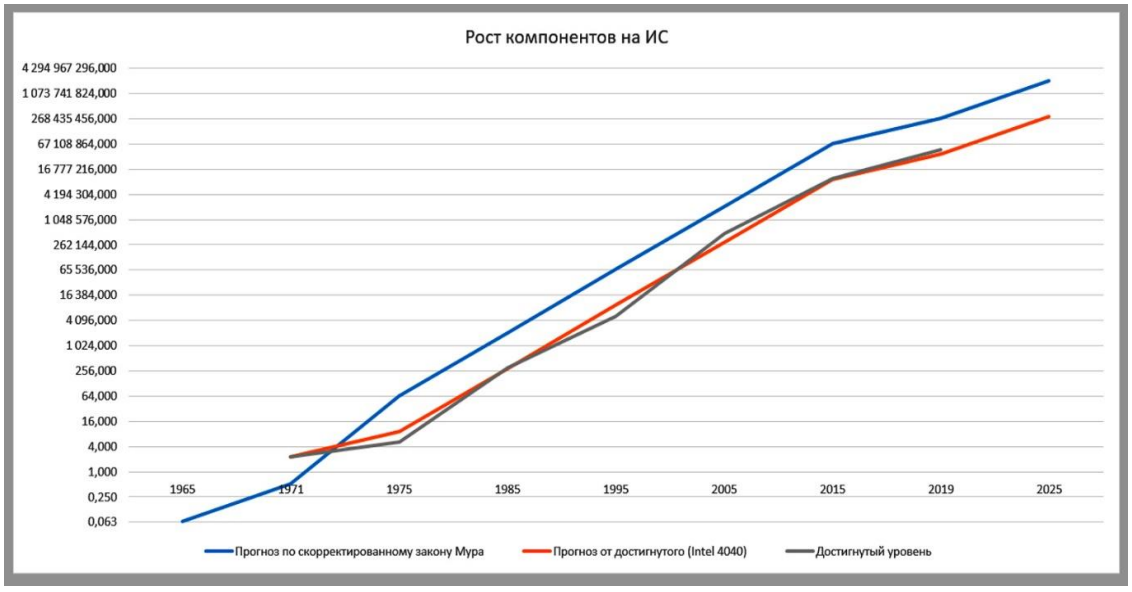



Рис. 4. График роста компонентов на ИС

Уменьшение размера транзисторов даёт возможность повысить их плотность на кристалле, увеличить их количество, соответственно, поднять вычислительные возможности микросхемы. Появляется возможность ввести в состав кристалла множество вычислительных ядер, позволяющих распараллелить решение различных задач и управлять обменом с внешними устройствами независимыми потоками данных.

Для решения задач многопоточности в состав кристалла интегрируется набор контроллеров высокоскоростных интерфейсов для связи и обмена информацией с соседними устройствами в виде сложно-функциональных блоков (СФ-блоки), таких как RapidIO, SATA, PCIe, USB, Gigabit Ethernet, DDR и другие. Конкретный набор СФ-блоков для каждой разрабатываемой микросхемы определяется исходя из технического задания. Схема построение микросхемы, когда в одном кристалле размещаются все необходимые контроллеры и процессорные ядра, принято называть системой на кристалле (СнК).

## 4. Переход на систему в корпусе

Обратной стороной миниатюризации элементов кристаллов является возрастание стоимости производства пластин по уменьшенным технологическим нормам. Например, стоимость первоначального запуска новых пластин с кристаллами по технологии 28 нм у одного из ведущих производителей (фабрика TSMC, Тайвань) превышает $1 млн. Запуск пластин по технологии 12 нм и ниже будет обходиться значительно дороже.

Соответственно возрастает цена ошибки в разрабатываемой топологии кристалла. Наличие ошибки в одном из СФ-блоков может привести к неработоспособности всей микросхемы и напрасно потраченным ресурсам, опозданием с выходом на рынок. Таким образом, по мере усложнения технологии изготовления кристаллов СнК сталкивается с нарастающими сложностями. Решением этой проблемы стала дезинтеграция кристалла, т.е. переход от «системы на кристалле» к «системе в корпусе» (СвК).

Технология СвК– это комбинация в одном модуле сразу нескольких активных и пассивных электронных компонентов, выполняющих разные функции. На сегодняшний день наиболее интересным и освоенным направлением СвК получило направление, называемое чиплет (chiplet). Чиплет – это кристалл, специально разработанный для функционирования с другими подобными кристаллами на одной подложке. На рисунке 5 приведен пример процессора фирмы

AMD Ryzen Threadripper, выполненный по технологии чиплетов [4].


Рис. 5. Процессор фирмы AMD Ryzen Threadripper, выполненный по технологии чиплетов

На рисунке 6 AMD представляет следующий шаг фирмы AMD по развитию своего чиплетного процессора [4]. Происходит дальнейшая дезинтеграция больших кристаллов. В отдельные чиплеты выделяются центральные процессоры, изготавливаемые по технологии 7 нм. В результате их количество возросло до 8 штук. Память и контроллеры ввода/вывода выделены в отдельный чиплет, выполненный по более дешевой технологии 14 нм.

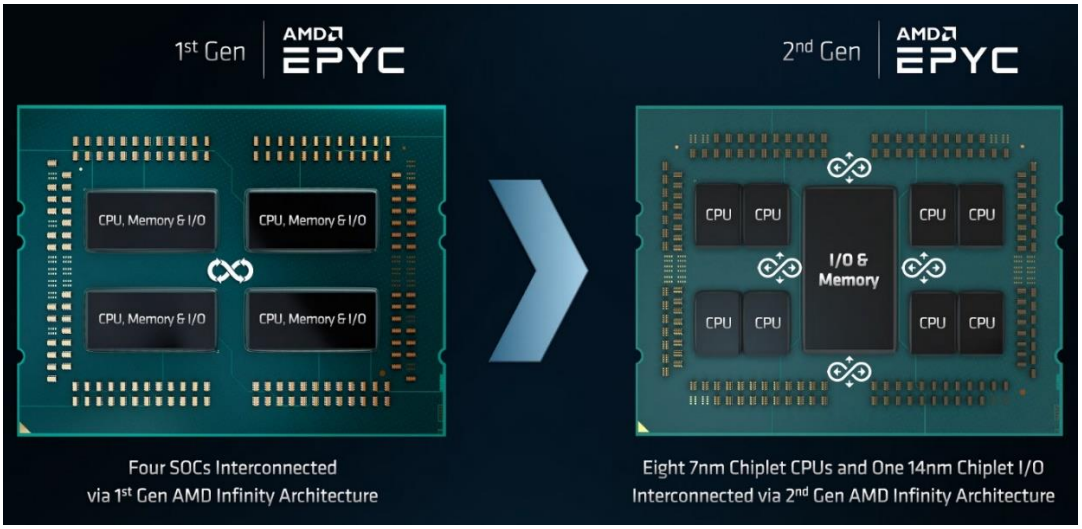



Рис. 6. Развитие чиплетного процессора фирмы AMD Ryzen Threadripper

Технология чиплетов показывает удивительную гибкость:

- можно умножать количество процессорных ядер при помощи чиплетов до необходимого количества для решения поставленной задачи;
- можно одновременно использовать разные чиплеты на базе передовых и предыдущих технологий, что даёт выигрыш в цене и надёжности микросхемы;
- можно одновременно использовать чиплеты, выполненные по разным технологиям - Si, GaAs, SiGe и др;
- можно одновременно использовать как цифровые чиплеты, так и аналоговые, разнообразные сенсоры, радиочастотные тракты, микроэлектромеханические системы (МЭМС) и т. д., изготовленные по наиболее подходящим для них технологиям.

Дополнительные преимущества чиплетов:

- использование небольших чиплетов заметно снижает уровень брака по сравнению с большими кристаллами СнК, что снижает

стоимость конечного продукта;

- уменьшение размеров кристаллов на пластине позволяет повысить заполняемость на краях пластины по сравнению с крупными кристаллами, что положительно сказывается на цене кристаллов.

## 5. Заключение

В статье приведены особенности полимерных и LTCC подложек для перевёрнутых кристаллов. Описан опыт в области разработки и эксплуатации микросхем в ФГУ ФНЦ НИИСИ РАН. Обозначены тенденции развития технологии производства кристаллов. Обоснован переход производителей сложных микросхем от «системы на кристалле» к многокристальной «системы в корпусе». Представлены преимущества применения чиплетов в сложных микросхемах.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2024-0003 "Методы разработки аппаратно-программных платформ на основе защищенных и устойчивых к сбоям систем на кристалле и сопроцессоров искусственного интеллекта и обработки сигналов".

# Challenges and Directions of Development of Microchips with a Flip-Chip Technology

**A.M. Baranov, A.A. Podkovyrov, A.V. Andreev**

**Abstract.** The paper discusses the challenges and directions of development of microchips with a flip-chip technology. The features and properties of polymer and LTCC substrates are considered. The main characteristics of microchips developed at the SRISA RAS are presented. The "System-in-Package" technology and its advantages are described.

**Keywords:** Flip-Chip technology, LTCC packages, organic packages, System-in-Package

## Литература

1. Баранов А.М., Подковыров А.А., Андреев А.В. Опыт НИИСИ РАН по разработке подложек для микросхем с перевернутым кристаллом. Труды НИИСИ РАН. 2023. Т. 13. № 3. С. 30-35.

2. Гусева М.А. Циановые эфиры – перспективные термореактивные связующие (обзор) // Авиационные материалы и технологии. 2015. №2 (35). С. 45-50.

3. Как умирал Закон Мура или эволюция транзисторов: от 2D к 3D. URL: https://club.dns-shop.ru/blog/t-100-protsessoryi/73291-kak-umiral-zakon-mura-ili-evolutsiya-tranzistorov-ot-2d-k-3d/. (дата обращения 12.02.2024)

4. Что такое чиплет? Преимущества и недостатки чиплетного дизайна. URL: https://club.dns-shop.ru/blog/t-100-protsessoryi/71441-chto-takoe-chiplet-preimuschestva-i-nedostatki-chipletnogo-dizaina/ (дата обращения 15.02.2024)

# Влияние случайных флуктуаций легирующей примеси на характеристики полевых кремниевых GAA нанотранзисторов

**Н.В. Масальский**[1]

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

**Аннотация.** Исследованы эффекты случайных флуктуаций легирующей примеси на электро-физические характеристики кремниевых полевых GAA нанотранзисторов с различными радиусами рабочей области. Показано, что транзисторы с меньшим радиусом характеризуются снижением среднего значения и вариации подпорогового наклона и DIBL-эффекта, тем самым повышая устойчивость к проявлению коротко-канальных эффектов. Напротив, относительные вариации тока стока транзистора с уменьшением диаметра возрастают, что связано со снижением проводимости рабочей области при более узких ее поперечных сечениях. Абсолютные флуктуации тока стока существенно зависят от количества примеси, проникающей в рабочую область. Для определения причин флуктуации токов стока кремниевых полевых GAA нанотранзисторов, изучены статистические характеристикам последовательного сопротивления истока/стока и низко полевой подвижности. Эти параметры критически одновременно влияют на разброс тока стока транзистора. Для нивелирования влияния механизма диффузии легирующей примеси в рабочую область рекомендуется ограничить уровень легирования областей сток/исток и использовать относительно большие из диапазона возможных поперечные сечения рабочей области. Это позволит обеспечить стабильные электро-физические характеристики транзисторов с высоким парированием коротко-канальных эффектов.

**Ключевые слова:** кремниевый нанотранзистор с полностью охватывающим затвором (GAA), флуктуация легирующей примеси, флуктуация проводимости и подвижности, разброс тока стока

## 1. Введение

Большой статистический разброс электро-физических характеристик из-за случайных флуктуаций примеси (random doping fluctuation (RDF)) является одной из основных проблем в наноразмерных полупроводниковых приборах [1-3]. В данном контексте рассматривают диффузию легирующей примеси высоколегированных областей истока/стока в качестве основного механизма ее инжекции в рабочую область (область канала). Это является наиболее важным, поскольку на фоне низколегированной рабочей области отличие уровней легирования рабочей области и областей стока и истока составляет более четырех порядков. Этот недостаток присущ также кремниевым полевым транзисторам с полностью охватывающим затвором (gate-all-around (GAA)). Эффекты RDF изменяют его электро-физические характеристики, включая пороговое напряжение (Uth), последовательные сопротивления истока/стока (Rsd) и проводимость (Gm), что связано с эффективной длиной канала и подвижностью носителей.

Необходимость исследования эффектов RDF обусловлена тем, что сложилась тенденция развития наноразмерных полевых транзисторов с низколегированными областями, которые используются в приложениях с низким энергопотреблением. При этом рабочий ток транзистора (ток Ion) снижается, но не до критического уровня [4, 5]. Для выполнения условий применимости важно исследовать влияние сочетания механизма RDF и вариативности топологических размеров GAA транзисторов на стабильность его электро-физических характеристик. Это позволяет уяснить причины разброса электро-физических характеристик определенной конструкции транзистора, найти способы для уменьшения вариаций характеристик устройства и сформулировать рекомендации для разработки конструкции устройства с высоким SOA. В настоящей работе мы исследовали совместное влияние эффектов RDF, индуцируемых высоколегированными областями истока/стока и флуктуации радиуса кремниевого GAA нанотранзистора с длиной канала близкой к современной отечественной технологии на вольт-амперные характеристики (ВАХ), в частности, пороговое напряжение и ток стока, и их производные. В настоящей работе не обсуждается механизм диффузии примеси из высоколегированных областей истока и стока в низколегированную рабочую область. Мы рассматриваем случай фиксирован-

ного количества инжектированной примеси, которая случайным образом распределяется по всему объему рабочей области.

## 2. Конструкция и параметры прототипа

Все прототипы кремниевого полевого GAA нанотранзистора были n-типа, имели одинаковые профили легирования областей сток/исток с пиковым значением Nds у внешнего края этих областей. Все рабочие области (области канала) легированы бором с фиксированным уровнем концентрации $N_A$. Низкий уровень легирования рабочей области отвечает современным технологическим требованиям [1, 5] и позволяет получить достоверную оценку влияния эффектов RDF на вариативность ВАХ кремниевых полевых GAA нанотранзисторов. Прототипы имели фиксированную длину затвора Lg и толщину подзатворого оксида tox – оксида кремния, диапазон диаметра рабочей области ( выбран от 7.5 до 12 нм. Геометрическое представление и значения вышеуказанных параметров приведены на рис. 1 и в табл. 1.

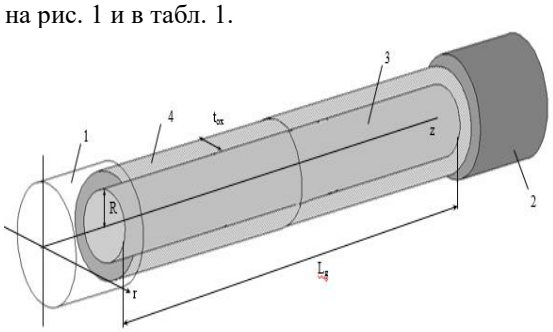



Рис. 1. Схема кремниевого GAA нанотранзистора, где 1 – исток, 2 – сток, 3 - кремниевая рабочая область, 4 - подзатворный окисел

Таблица 1. Параметры прототипа транзистора

| Параметр | значение |
|---|---|
| Lg, нм | 22 |
| tox, нм | 1.2 |
| $N_{da}$,см$^{-3}$ | 5.0x10$^{19}$ |
| $N_A$,см$^{-3}$ | 1.0x10$^{15}$ |

## 3. Алгоритм моделирования

Для эффективного исследования вариаций вольт–амперные характеристики (ВАХ) кремниевых полевых GAA нанотранзисторов моделируются с использованием самосогласованного решения дрейфа-диффузионного уравнения в сочетании с уравнением Пуассона. Предложенный подход был апробирован при моделировании нескольких типов кремниевых GAA транзисторов и во всех случаях достигалось требуемое согласование с экспериментальными данными [6 и библиография к ней].

Для включения механизма RDF в базовую модель кремниевого полевого GAA нанотранзистора был использован подход изложенный в [7, 8]. Поскольку все прототипы имели низкую концентрацию примеси в рабочей области, рандомизированы были только примеси инжктированные из областей сток/исток в соответствии с кинетическим методом Монте-Карло [9-11]. В нашем случае упрощено рассматриваются дальнодействующие и короткодействующие кулоновские потенциалы. Коэффициент экранировки был выбран $2N^{1/3}$, где N – концентрация инжектированной примеси (мышьяка). Размещение отдельных атомов мышьяка в рабочей области в виде точечных потенциалов, что иллюстрируется рис.2, упрощает процесс моделирования дрейфа-диффузионного переноса носителей [12]. При этом адекватность модели не снижается.

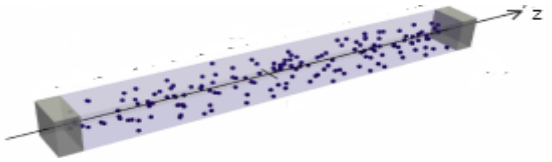



Рис. 2. Схематичное представление рабочей области [12]. Инжектированная примесь изображена точками и распределена случайным образом.

Количество выборок для исследования RDF составляло 100 для каждого значения диаметра рабочей области с шагом 0.5 нм. Линейные токи стока Ids вычислялись при управляющих напряжения на стоке Uds 0.05 В и затворе Ugs 0.75 В. Максимальные значения электропроводности Gm,max были определены при Uds 0.05 В. Значения порогового напряжения Uth в линейном режиме (Uth,lin) были извлечены с использованием метода постоянного тока.

Чтобы проанализировать эффекты RDF, мы исследовали характерное поведение следующих параметров – сопротивление стока/истока Rsd и низко-полевая подвижность. Эти параметры имеют электрическую природу, связанную с эффективностью транспорта носителей через транзистор. Следует отметить, что значение параметра Rsd рассчитывается с использованием Y-функции. Уравнения Y-функции для извлечения Rsd в линейном режиме приведено в [13]. В ходе исследований анализируется характеристика –

чувствительность. Чувствительность определяется как наклон диаграмм рассеяния, который представляет собой разброс исследуемого параметра, например, тока стока транзистора, по отношению к флуктуациям параметров подверженных влиянию механизма RDF. Чувствительность показывает, насколько сильно отклонения этих параметров влияют на вариативность тока.

## 4. Результаты и обсуждение

Из полученных результатов подпороговом режиме можно извлечь важные характеристики - подпороговый наклон (SS) и DIBL-эффект для различных значений радиусов рабочей области. Значения параметра SS извлекаются из ВАХ ниже порогового напряжения Uth при Uds =0.8 В. В нашем случае значения параметра DIBL вычисляются из соотношения [14]: $DIBL = -(U_{th,sat} - U_{th,lin})/(U_{ds,sat} - U_{ds,lin})$ , где Uth,sat - это пороговое напряжение, извлеченное при Uds= 0.75 В, Uds,sat = 0.75 В, Uds,lin=0.05 В.

На основе полученных результатов можно сделать следующие выводы. Снижение уровня побочной примеси в канале кремниевого GAA нанотранзистора является положительным фактором, т.к. это приводит к снижению средних значений параметров SS и DIBL, не смотря на то что с ростом радиуса значения параметров DIBL и SS возрастают. Однако, это увеличение незначительно и составляет менее 10% в исследуемом диапазоне радиусов рабочей области. И хотя значения SS и DIBL достаточно большие, стандартные отклонения параметров SS и DIBL, напротив, по сравнению с ними, невелики, что иллюстрируется рис. 3, несмотря на то, что поведение зависимостей $\sigma(SS)$ и $\sigma(DIBL)$ разное.

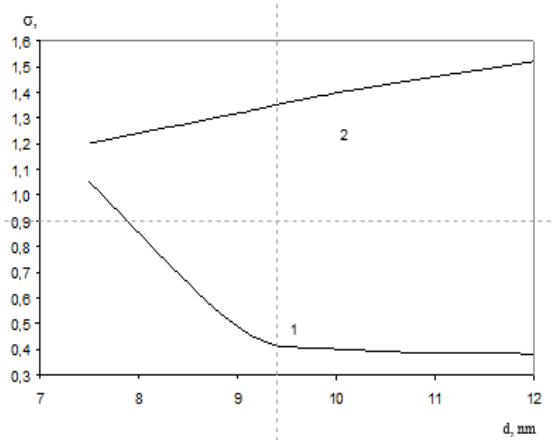



Рис. 3. Зависимость стандартного отклонения SS (кривая 1) и DIBL (кривая 2) от диаметра рабочей области

Зависимость $\sigma(SS)$ имеет ярко выраженный нелинейный убывающий характер, зависимость $\sigma(DIBL)$ , напротив, практически монотонно возрастает. При этом максимальные стандартные отклонения $\sigma_{\max}(SS)$ и $\sigma_{\max}(DIBL)$ равны 2,8 мВ/дек и 3 мВ/В.

Анализируя вариации параметра Uth,lin, можно сделать аналогичный вывод о том, что они, обусловленные эффектами RDF, невелики. В данном случае максимальное стандартное отклонение составляет 4.7 мВ. Эта величина оказывается значительно меньше теплового потенциала и сопоставима с тем же действием, что оказывает на параметр Uth,lin наличие высоколегированныи рабочих областей [15]. По совокупности, можно утверждать, что ключевые характеристики подпорогового режима кремниевых полевые GAA нанотранзисторов с низко легированной рабочей областью невосприимчивы к проявлению эффектов RDF.

Чтобы исследовать причины вариативности тока стока Id,lin, проанализированы его взаимодействия со флуктуациями порогового напряжения $U_{th,lin}$ и проводимости Gm,max. По методике, изложенной в [8] были рассчитаны коэффициенты корреляции между ними. Результаты расчётов приведены на рис. 4.

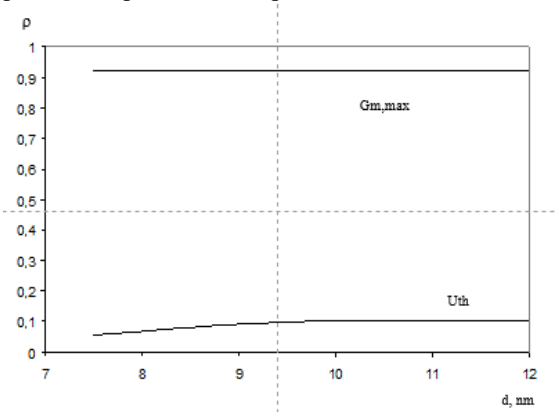



Рис. 4. Зависимость коэффициента корреляции $\rho$ от диаметра рабочей области

Из полученных данных следует, вариативность тока Ids определяется изменениями параметра Gm,max. При этом компоненты, которые могут порождать его флуктуации достаточно ограничены. К ключевым можно отнести следующие параметры: Lg, d, tox, mu0 и Rsd. В этом списке присутствуют да новых параметра- mu0 – низко полевая подвижность и Rsd - последовательное сопротивление стока и истока. Однако, только эти параметры характеризуются выраженной зависимостью от эффектов RDF.

На рис. 5 приведены экстрагированные зависимости вариации значений параметров Ids и Gm,max в исследуемом диапазоне радиусов рабочей области R.

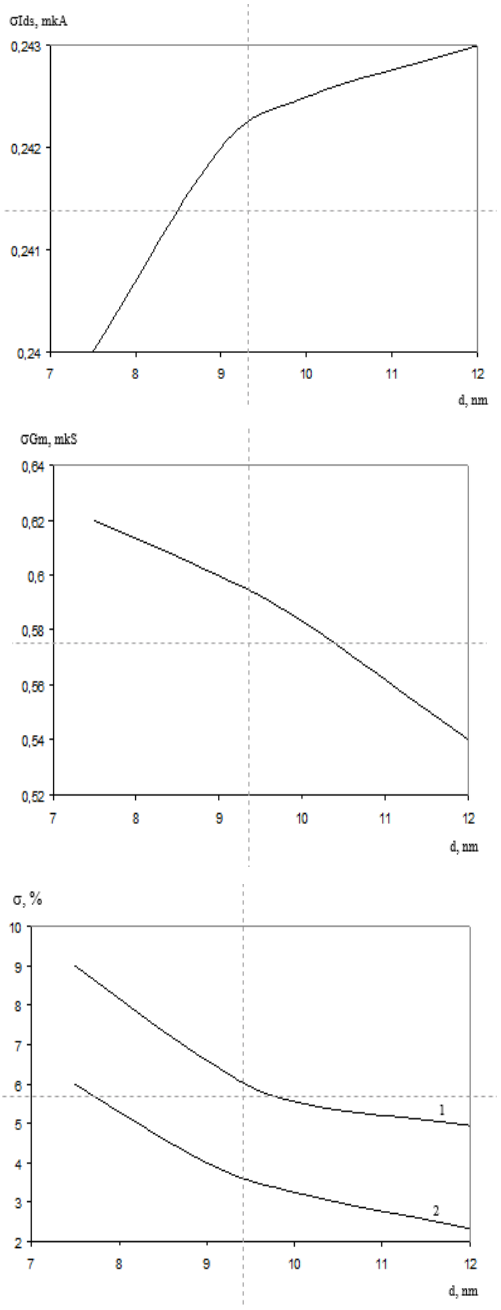
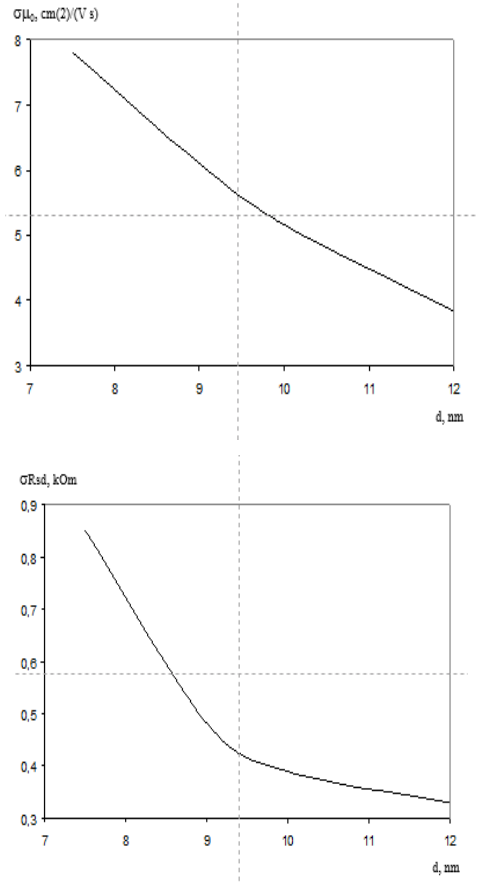



Рис. 5. Зависимость вариации тока Ids (верхний рисунок) и Gm,max (средний рисунок) от d и их представление в процентном отношении (нижний рисунок), где 1 - $\sigma Ids(d)$ , 2 - $\sigma Gm,\max(d)$

Из приведённых результатов следует, что флуктуации тока Ids незначительно увеличиваются с ростом радиуса рабочей области или со снижением концентрации инжектированной примеси. Абсолютные вариации параметра Gm,max демонстрируют противоположное поведение – с ростом они монотонно снижаются. Причем более значимо, чем вариации тока. В процентном выражении – более 10% для параметра Gm,max и менее 1% для Ids. Следует отметить что, с ростом R относительные вариации исследуемых параметров практически пропорционально снижаются и не зависят от уровня концентрации побочной примеси. Средние значения тока Ids и проводимости Gm,max коррелируют с возрастанием/снижением параметра R из-за проявления ККЭ, которые определяют управляемость затвора протекания тока через транзистор. Следует отметить, по мере уменьшения R относительная флуктуация тока Ids увеличивается из-за общей тенденции снижения среднего значения. То же свойство присуще и параметру Gm,max.

На рисунке 6 представлены зависимости вариации параметров Rsd и mu0 (которые были определены выше) от радиуса рабочей области.



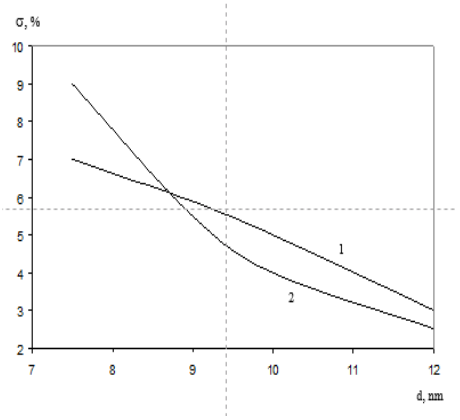



Рис. 6. Зависимость вариации подвижности mu0 (верхний рисунок) и сопротивления Rsd (средний рисунок) от d и их представление в процентном отношении (нижний рисунок), где 1 - $\sigma Rsd(d)$,

2 - $\sigma\mu 0(d)$

В отличие от поведения аналогичной зависимости для тока Ids абсолютные значения вариаций Rsd и mu0 возрастают по мере уменьшения параметра d. При этом зависимости sigm(mu0) и sigm(Rsd) имеют разную крутизну. Однако, изменение величины стандартного отклонения по рассматриваемому диапазону d составляет примерно 50%. Это позволяет сделать вывод об одинаковом влиянии вариаций параметров Rsd и mu0 на вариации тока Ids. Относительные вариации Rsd и mu0 следуют той же тенденции, что характерна для относительных флуктуаций параметров Ids, lin и Gm,max. Отметим, что их диапазон составляет примерно одинаковое значение процентов. Такое совпадение обосновано тем, что по мере уменьшения радиуса рабочей области происходит одинаковое возрастание средних значений этих параметров. В совокупности с ростом их абсолютных вариаций поддерживается диапазон значений относительных вариаций от 3% до 9%. Следует отметить, что снижение концентрации инжектированной примеси в рабочей области положительным образом влияет на флуктуации параметра mu0. А вариации Rsd напротив увеличиваются из-за возможно больших расстояний между отдельными частицами легирующей примеси мышьяка.

На рис. 7 показаны диаграммы разброса тока Ids относительно параметров Rsd и mu0.

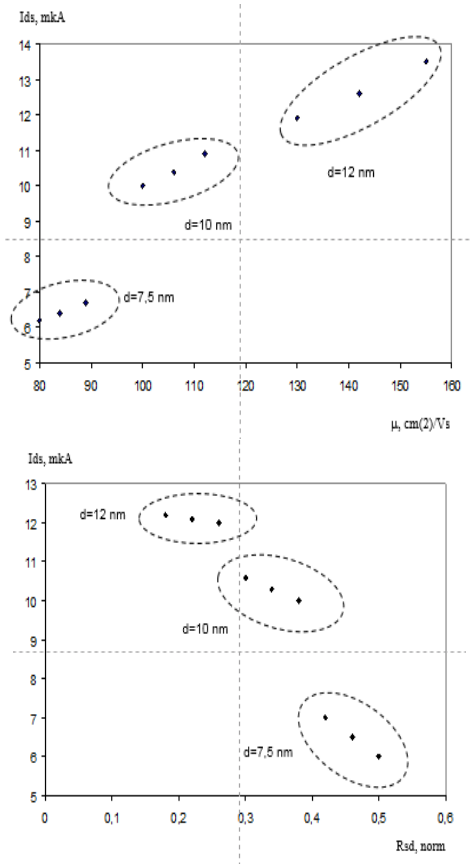



Рис. 7. Диаграммы разброса тока Ids-mu0 (верхний рисунок) и Ids-Rsd нижний рисунок

Вариативность параметра Ids сильно коррелирует с характерным поведением параметров, имеющих электрическую природу. В нашем случае это Rsd и mu0. Из представленных данных следует, что флуктуации Rsd и mu0 согласовано влияют на абсолютные отклонения параметра Ids. Выше было показано, что снижение концентрации инжектированной примеси в рабочей области существенно снижает флуктуации mu0 по сравнению с ростом разброса параметра Rds, однако, что следует из рис. 7, флуктуации тока Ids уменьшаются. Из приведенных данных следует, что по мере роста величины R чувствительность существенно снижается. Это приводит к уменьшению влияния флуктуаций параметров Rsd и mu0 на вариативность тока lds. Снижение управляемости по току lds при малых R приводит к небольшим относительным отклонениям параметра Ids.

Абсолютные флуктуации параметра Gm,max зависят от вариативности параметров Rds и mu0. С уменьшением параметра R концентрация инжектированной примеси в рабочей области возрастает. Это приводит к росту флуктуаций параметра mu0 и, следовательно, Gm,max, что иллюстрируется рис. 7. И наоборот, увеличение радиуса рабочей области снижает разброс параметра

mu0, но усиливает влияние вариативности параметра Rsd. Таким образом, конкурирующая зависимость между вариациями Rds и mu0, во-первых, снижает вариативность параметра Gm,max по сравнению, параметром Ids, во-вторых, согласованным выбором радиуса рабочей области и концентрации индуцированной примеси (в нашем случае мышьяка) можно минимизировать флуктуации параметра Gm,max.

## 5. Заключение

Разработан способ моделирования флуктуаций электрофизических характеристик кремниевых полевых GAA нанотранзисторов с учетом RDF механизма, индуцируемых высоколегированными участками источника/стока. При помощи численного 3D-моделирования было проведено исследование влияния RDF-эффектов на характеристики GAA нанотранзисторов с фиксированной длиной и изменяющемся радиусом рабочей области. Показано, что вариативность тока транзистора и его проводимости всецело определяются флуктуациями подвижности и последовательного сопротивления. Вариативность порогового напряжения не коррелирует с разбросом тока и проводимости. Флуктуации ряда ключевых параметров, в частности порогового напряжения и подпорогового наклона, невелики и практически не зависят от радиуса рабочей области. Установлено, что конкурирующая зависимость между вариациями электрическими параметрами снижает флуктуации проводимости по сравнению с разбросом тока Ids, а согласованным выбором радиуса рабочей области и концентрации индуцированной примеси можно минимизировать флуктуации проводимости. Если минимизировать инжекцию легирующих примесей из областей стока и истока в рабочую область транзистора при максимально возможном радиусе рабочей области, то возможно решить две задачи: снизить флуктуации тока и проводимости, а также уменьшить влияние коротко-канальных эффектов и повысить защиту от пробоя транзистора.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2024-0003 "Методы разработки аппаратно-программных платформ на основе защищенных и устойчивых к сбоям систем на кристалле и сопроцессоров искусственного интеллекта и обработки сигналов".

# The Effect of Random Fluctuations of a Doping Impurity on the Characteristics of Field-Effect Silicon GAA Nanotransistors

**N. Masalsky**

**Abstract.** The effects of random fluctuations of an alloying impurity on the electrophysical characteristics of silicon field GAA nanotransistors with different radii of the working area are investigated. It is shown that transients with a smaller radius are characterized by a decrease in the average value and variation of the subthreshold slope and the DIBL effect, thereby increasing resistance to short-channel effects. On the contrary, the relative variations of the transistor drain current increase with decreasing diameter, which is associated with a decrease in the conductivity of the working area with narrower cross-sections. The absolute fluctuations of the flow current depend significantly on the amount of impurity penetrating into the working area. To determine the causes of fluctuations in the flow currents of silicon field GAA nanotransistors, statistical characteristics of the source/drain series resistance and low field mobility were studied. These parameters critically affect the current spread of the transistor drain at the same time. To offset the influence of the diffusion mechanism of the alloying impurity into the working area, it is recommended to limit the level of doping of the drain/source areas and use relatively large cross-sections of the working area from the range of possible ones. This will ensure stable electro-physical characteristics of transistors with high parry of short-channel effects.

**Keywords:** silicon nanotransistor with surrounding gate (GAA), fluctuation of dopant, fluctuation of conductivity and mobility, flow current drain silicon

## Литература

1. More Moore. International Roadmap for Devices and Systems. IRDS, Piscataway, NJ, USA, 2021
2. N Sano, K. Yoshida, G. Park. Fundamental aspect of semiconductor device modeling associated with

discrete impurities: drift-diffusion scheme. “IEEE Trans. Electron Devices”, (2020), vol. 67, 3323-3328.

3. Н.В. Масальский. Чувствительность распределения потенциала конических GAA нанотронзисторов к вариациям топологических размеров рабочей области, “Труды НИИСИ РАН”, (2023), т. 13(3), 23-29

4. B. D. Gaynor, S. Hassoun. Fin shape impact on FinFET leakage with application to multithreshold and ultralow-leakage FinFET design. “IEEE Trans. Electron Devices”, (2014), vol. 61, 2738–2744.

5. M. V. Fischetti, W. G. Vandenberghe. Advanced Physics of Electron Transport in Semiconductors and Nanostructures, New York, U.S.A.: Springer, 2016.

6. Масальский Н.В. Моделирование ВАХ ультра тонких КНИ КМОП нанотранзисторов с полностью охватывающим затвором.“Микроэлектроника”, (2021), т. 50, 436-444.

7. K. Huang, Statistical Mechanics, 2nd ed. New York, U.S.A.: John Wiely& Sons, 1987.

8. Kubo, M. Toda, N. Hashitsume. Statistical Physics II: Nonequilibrium Statistical Mechanics, 2nd ed. Berlin, Germany: Springer, 1991.

9. K. Nakanishi, T. Uechi, N. Sano. Self-consistent Monte Carlo device simulations under nano-scale device structures: Role of Coulomb interaction, degeneracy, and boundary condition. “Technical Digest. Int. Electron Device Meeting”, (2009), Dec 2009, 1–4.

10. C. Jacoboni. Theory of Electron Transport in Semiconductors: A Pathway from Elementary Physics to Nonequilibrium Green Functions. New York, U.S.A.: Springer, 2010.

11. M. Uematsu, K. M. Itoh, G. Mil’nikov, H. Minari, N. Mori. Simulation of the effect of arsenic discrete distribution on device characteristics in silicon nanowire transistors. “Tech. Dig. Int. Electron Devices Meet.”, (2012), 709–712.

12. N. Sano, K. Matsuzawa, M. Mukai, N. Nakayama. On discrete random dopant modeling in drift-diffusion simulations: physical meaning of ‘atomistic’ dopants. “Microelectron. Reliab.”, (2002), vol. 42, 189–199.

13. G. Tomar, A. Barwari. Fundamental of electronic devices and circuits. Springer, 2019.

14. M. Lundstrom, J. Guo. Nanoscale Transistors: Device Physics, Modeling and Simulation. Springer: New York, 2006.

15. J.-P. Colinge, FinFETs and Other. Verlag, New York, NY, USA, 2008.

# Повышение равномерности псевдослучайных чисел

**А.С. Куцаев**[1]

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

**Аннотация.** Качество выработки случайных тестов зависит от выбора генератора случайных чисел. Проверка популярных генераторов с помощью критерия Пирсона хи-квадрат показывает, что равномерное распределение с уровнем значимости 70% и выше наблюдается лишь у половины выборок. По этой причине, чтобы получить необходимое покрытие, нужно увеличивать объем тестов. Предлагается способ повышения равномерности распределения для генераторов случайных чисел, основанный на фильтрации выборки. Подбор параметров фильтра позволяет получить нужную равномерность распределения при умеренном числе пропусков.

**Ключевые слова:** генератор случайных чисел, равномерное распределение, хи-квадрат, фильтрация, случайные тесты.

## 1. Введение

В основе компьютерных датчиков псевдослучайных чисел (ПСЧ) обычно лежит моделирование равномерного распределения целых чисел на отрезке от нуля до наибольшего при данной разрядности. Уже первые опыты с датчиками показали, что сложные манипуляции с числами не гарантируют качество результата. Примером может служить метод "середины квадрата" Д. фон Неймана. Это рекурсивный алгоритм, в котором очередное число получается из предыдущего возведением его в квадрат и вырезанием отрезка битов из средней части результата. При этом количество единичных битов результата обычно снижается, и с какого-то момента датчик выдает только нули.

В настоящее время предложено много методов генерации ПСЧ, как универсальных, так и специализированных. Разрядность датчиков меняется от 32 до 128 битов. Генераторы большой разрядности используются в системах с повышенными требованиями к безопасности. Датчик ПСЧ должен моделировать равномерность распределения и случайность значений в выборках. Для компьютерного применения потребовались также минимум затрат, возможность точного воспроизведения уже полученной выборки, а затем и криптографическая стойкость, т.е. защита от попыток восстановить случайную последовательность по ее части.

Один из популярных способов получения ПСЧ предложен в 1948 году Д.Х. Лемером [1]. Рекурсивная схема основана на выражении $x_{n+1} = (a \cdot x_n + c)_{mod\ m}$, где a, c, m - константы. Полученная последовательность ПСЧ называется линейной конгруэнтной последовательностью. Она периодическая, и величина периода отчасти характеризует ее случайность. Разработаны правила для выбора констант $a$, $c$, $m$, но их соблюдение не гарантирует высокого качества выборок.

Другое семейство методов под общим названием XorShift предложено Д. Марсалья в 2003 году [2]. В его основе набор операций сдвига и исключающего ИЛИ над одним или несколькими предыдущими членами последовательности ПСЧ.

Из более сложных методов нужно отметить Mersenne Twister, разработанный М. Мацумото и Т. Нисимура в 1997 году и также имеющий несколько известных реализаций [3], [4]. Его особенностью среди прочего является "размазывание" результатов арифметических операций по достаточно длинному (до 624 слов) буферу, что позволяет получить огромный период датчика (для MT19937 это $2^{19937}$ - 1). Кроме того, после применения рекуррентного соотношения делается перемешивание битов результата, названное "закалкой" (tempering). Дальнейшее усложнение Mersenne Twister, названное WELL (Well Equidistributed Long-period Linear, [5]), позволило повысить недостаточную равномерность распределения.

При генерации случайных тестов часто используются небольшие выборки ПСЧ, равномерно распределенные на отрезке небольшой длины. Так, для случайного выбора аргументов инструкций требуется выбор 5-битового непосредственного значения или номера регистра, причем не все возможные значения допустимы. Неравномерность проявляется как частый выбор близких значений, и компенсировать ее можно только увеличением объема тестового материала. Равномерность здесь требуется в основном для наиболее значимых битов чисел.

В данной работе предлагается способ повышения равномерности распределения любого датчика ПСЧ с помощью фильтрации. В Главе 2

рассмотрена оценка равномерности для ряда популярных датчиков ПСЧ. В Главе 3 описан способ повышения равномерности распределения. В Главе 4 рассматриваются вероятностные оценки для получаемых выборок. Глава 5 содержит выводы. В Приложении приведены схемы использованных датчиков ПСЧ.

## 2. Оценки равномерности

Проверка закона распределения обычно начинается с оценок среднего, дисперсии и анализа частот с помощью критерия хи-квадрат [6]. В популярных датчиках среднее и дисперсия сходятся к теоретическим значениям, тогда как проверка равномерности не всегда дает желаемый результат.

При вычислении оценки критерия хи-квадрат для выборки из $n$ чисел область изменения значений разбивается на $k$ интервалов и находится сумма $\chi^2$:

$$\chi^2 = n \cdot \sum_{i=1}^{k} \frac{(n_i/n - p_i)^2}{p_i}, \qquad (2.1)$$

где $n_i$ - количество чисел выборки, попавших в интервал $i$, а $p_i$ - теоретическая вероятность попадания этот же интервал. Для равномерного распределения, если все интервалы одной длины, $p_i = 1/k$ и выражение для оценки можно упростить.

$$\chi^2 = \sum_{i=1}^{k} \frac{(n_i - n/k)^2}{n/k} \qquad (2.2)$$

Полученная оценка сравнивается с распределением хи-квадрат Пирсона с $k$ - 1 степенями свободы (так как сумма частот $n_i$ равна $n$, независимых частот на единицу меньше). Если значение оценки меньше, чем квантиль функции распределения хи-квадрат для некоторой вероятности $p$, то с этой вероятностью выборка соответствует заданному закону распределения (в данном случае - равномерному). Иначе либо закон распределения принимается с меньшей вероятностью, либо отвергается.

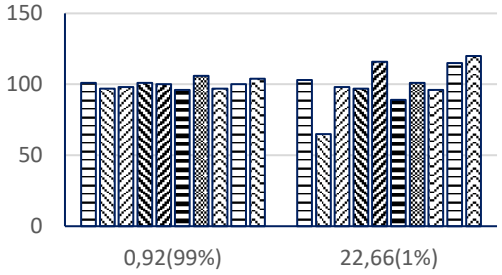



Рис. 1. Гистограммы выборок ПСЧ для крайних уровней значимости по хи-квадрат.

Для наглядности на Рис. 1 приведены гистограммы выборок ПСЧ для лучшего ($\chi^2$ = 0.92) и худшего ($\chi^2$ = 22.66) значений оценки на 10 интервалах при длине выборки 1000. Уровень значимости для равномерности составляет здесь 99% и 1%, соответственно.

Далее для сравнения были взяты линейный конгруэнтный генератор (LCG), две версии генератора XorShift и две версии Mersenne Twister. Алгоритмы и параметры генераторов подробно описаны в Приложении.

Датчики вырабатывают 32-битовые целые ПСЧ, равномерно распределенные на отрезке от 0 до $2^{31}$-1. Далее они преобразуются к более удобному полуинтервалу чисел с плавающей точкой [0, 1), который разбивается на $k$ одинаковых частей. При преобразовании использовались 23 наиболее значимых бита числа из 32.

Для оценки равномерности брались серии выборок, полученных при различных значениях "зерна". Значения $\chi^2$ у выборок менялись в широких пределах, а соответствующий уровень значимости для равномерности распределения мог принимать значения от 1 до 99%. Для более детального анализа можно рассмотреть гистограммы получаемых значений $\chi^2$ в серии выборок.

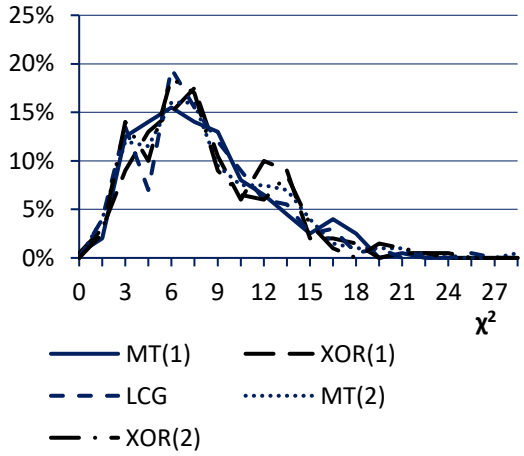



Рис. 2. Гистограммы значений $\chi^2$ по 10 интервалам на 200 выборках для датчиков Mersenne Twister (MT 1 и 2), XorShift (XOR 1 и 2), LCG.

На Рис. 2 показаны гистограммы $\chi^2$ для пяти датчиков ПСЧ. Длина выборок 2000, значение $\chi^2$ вычислялось по 10 интервалам на 200 выборках. Отрезок изменения $\chi^2$, общий для всех выборок, разбит на 20 частей, показан процент значений, попадающих в каждую часть.

У всех гистограмм значения медиан (от 6.4 до 7.1) отвечают уровню значимости около 70%. С ростом значений $\chi^2$ соответствующая доля выборок убывает, оставаясь ненулевой и для очень малых уровней значимости. Это значит, что мак-

симум значений $\chi^2$ не подходит в качестве критерия для оценки и сравнения качества датчиков. Также ни один датчик не дает гарантии равномерного распределения. Для оценки качества можно брать долю выборок, для которых уровень значимости по хи-квадрат превышает определенный порог, например, 80%. Для сравнения датчиков можно также использовать среднее значение $\chi^2$ по серии выборок.

## 3. Фильтрация

Испытанным приемом при генерации случайных чисел является пропуск части выработанных значений согласно заданному условию. Такие пропуски используются, например, при генерации случайных чисел с нормальным распределением либо для ускорения работы. Этот же подход можно использовать для повышения равномерности любого датчика ПСЧ.

Для фильтрации разобьем область значений ПСЧ на $F$ одинаковых интервалов. С каждым интервалом связан счетчик попавших в него чисел. Введем порог для разности максимума и минимума текущих значений счетчиков. Если очередное ПСЧ приводит к превышению порога, оно пропускается. Таким образом, значения всех счетчиков отличаются друг от друга на величину, не превышающую порог. Порог выбирается так, чтобы обеспечить нужную равномерность, а доля пропусков была бы не слишком велика.

Такая фильтрация выравнивает гистограмму выборки в случае, когда число интервалов гистограммы $k$ равно числу интервалов фильтрации $F$. Для других $k$ влияние фильтрации сложнее, но и в этом случае происходит определенное выравнивание гистограмм.

При выборе числа интервалов фильтрации $F$ и порога $d$ нужно учитывать, что фильтрация делает оценку хи-квадрат зависимой от размера выборки $n$. При $k=F$ величины $n_i$ в (2.1) совпадают со значениями счетчиков фильтрации. Можно показать, что при пороге фильтрации $|n_i - n_j| \le d$ и четном $k=F$ максимум суммы в $\chi^2$ достигается, если половина счетчиков имеет значение $n/k+d/2$, и половина $n/k-d/2$.

$$\chi^2 = \sum_{i=1}^{k} \frac{(n_i - n/k)^2}{n/k} \le \frac{k^2 d^2}{4n} \qquad (3.1)$$

В этом случае размер выборки существенно влияет на значение $\chi^2$. Например, при $k=F=16$, $d=10$ и $n=1000$ оценка (3.1) дает значение 6.4, что ниже квантиля распределения хи-квадрат с $r=15$ степенями свободы для уровня значимости 95%. Иначе говоря, при большом размере выборки оценка хи-квадрат покажет сколь угодно высокую равномерность выборки после фильтрации независимо от качества использованного датчика ПСЧ. Вообще неравенство (3.1) с заменой $k$ на $F$ выполняется при $Ak = F$ для любых целых $A$. Для других значений $k$ неравенство (3.1), вообще говоря, не выполняется, но ограничение фильтрации сказывается и здесь: с ростом размера выборки падает чувствительность оценки хи-квадрат.

Пример зависимости среднего значения $\chi^2$ от числа интервалов $k$ показан на Рис. 3. Использован датчик MT19937, фильтрация с порогом 10 при числе интервалов $F$=16 и $F$=32. Осреднение делалось по 200 выборкам, длина выборки 2000. Здесь же приведены квантили распределения хи-квадрат для уровня значимости P = 90%, а также изменение $\chi^2$ в отсутствие фильтрации ($F$ = 0).

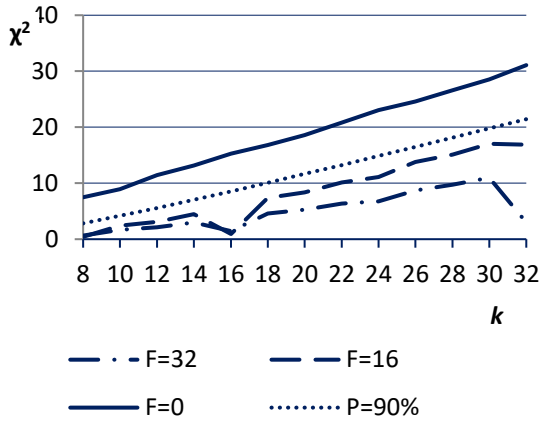



Рис. 3. Средние значения $\chi^2$ на 200 выборках для датчика MT19937 при фильтрации с $F$ = 32 и $F$ = 16 интервалами, а также без фильтрации ($F$ = 0), в зависимости от числа интервалов оценки хи-квадрат $k$. Для справки показаны квантили распределения хи-квадрат для уровня значимости $P$ = 90%.

На графиках $\chi^2$ видны провалы при $k = F$. Также есть провалы при $2k = F$ и $4k = F$, согласно замечанию для $Ak = F$ выше. Средние значения $\chi^2$ при наличии фильтрации лежат ниже графика квантилей для уровня значимости 90%, но максимумы в серии заметно выше этого графика. Поведение кривых показывает, что случаи, когда $F$ кратно $k$, не подходят для подбора параметров фильтрации. Для этого можно использовать осреднение по набору используемых на практике значений k, исключив из него случаи кратности.

При оценках равномерности нужно учитывать размер выборки $n$. Для больших размеров ($n$ = 2000 и выше) равномерность при фильтрации может завышаться, поскольку при вычислении оценки ограниченные величины делятся на $n$.

Это видно по провалам графиков на Рис. 3. Размеры $n$ = 100 и ниже не подходят, так как частоты попадания в интервалы оказываются меньше 5, что делает оценку хи-квадрат слишком грубой. Далее при оценках используются размеры выборки 200 и 400.

Подбор параметров фильтрации удобнее начать с анализа доли пропущенных значений, так как она зависит в основном от порога $d$ и числа интервалов фильтрации $F$. На Рис. 4 показана зависимость доли пропусков от порога при $n$ = 400. Для каждого числа интервалов $F$ (16 и 32) кривые от разных датчиков близки и сливаются, поэтому они заменены одной кривой, проведенной по середине полосы, содержащей исходные кривые. Отклонение не превышает 1.6% (в единицах оси ординат).

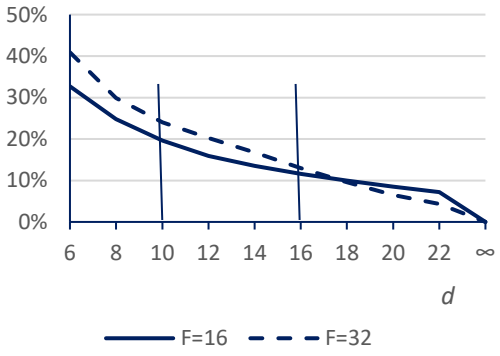



Рис. 4. Зависимость доли пропусков от порога при $F$=16 и 32.

Здесь видно, что приемлемые величины порога $d$ от 10 до 16: выше этих значений фильтрация мало влияет, а ниже слишком много отбрасывает. На этом отрезке сравнение кривых показывает, что при $F$=16 доля пропусков меньше.

Далее нужно выбрать тип датчика и число интервалов фильтрации. Оценки делались для всех типов датчиков по каждому числу интервалов хи-квадрат $k$ из набора {10, 12, 14, 18, 20, 22} и затем осреднялись. Основной интерес представляли значения порога $d$ от 10 до 16, выбранные выше. На этом отрезке датчик MT19937 дает лучшие результаты для размеров выборки 200 и 400 при $F$=16 и $F$=32, причем при $F$=16 равномерность выше.

Рис. 5 показывает повышение равномерности распределения за счет фильтрации с выбранными параметрами $F$=16 и $d$=10. Здесь показана зависимость доли приемлемых выборок от числа интервалов критерия хи-квадрат $k$. Приемлемыми считаются выборки, у которых оценка $\chi^2$ отвечает равномерному распределению с уровнем значимости 90%. Результаты получены на 500 выборках, длина выборки 200 и 400.

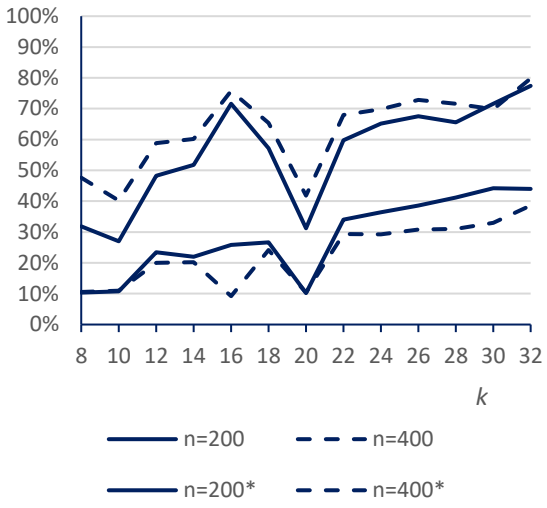



Рис. 5. Зависимость доли приемлемых выборок от числа интервалов в оценке хи-квадрат для датчика MT19937 на 500 выборках, длина выборки 200 и 400. Две верхние кривые с фильтрацией при $F$=16 и $d$=10. Две нижние кривые (отмечены *) без фильтрации.

На Рис. 5 видно, что фильтрация повышает долю выборок с равномерным распределением, оцениваемую по критерию хи-квадрат, более чем вдвое. Результат обычно зависит от исходной степени равномерности. Кроме того, влияет специфика цифровой генерации ПСЧ: все датчики во всех случаях дают ухудшение оценок при $k$=20. Это может быть связано с особенностями перемешивания битов в датчиках.

## 4. Вероятностные оценки

Оценки влияния фильтрации позволяют убедиться, что фильтрация не ухудшает свойства выборок (помимо равномерности), существенные при генерации тестов. В примерах использовался датчик MT19937.

Пусть датчик ПСЧ моделирует непрерывную случайную величину X, равномерно распределенную на полуинтервале [-1/2, 1/2). Разобьем полуинтервал на $k$ одинаковых частей. Для оценки влияния фильтрации выберем $k$ равным числу интервалов фильтрации $F$. Значения X представим в виде $x = y + z$, где

$$y = \left[\left(x + \frac{1}{2}\right) \cdot k\right] / k - \frac{1}{2}, \quad z = x - y \quad (4.1)$$

Здесь квадратные скобки обозначают целую часть. Будем рассматривать $y$ и $z$ как значения случайных величин Y и Z, соответственно. Дискретная случайная величина Y принимает значения $Y_j$, $j$ = 1, 2,... $k$. В разбиении полуинтервала

[-1/2, 1/2) на равные части $Y_j$ являются координатами середин частей. Значения непрерывной случайной величины Z лежат на полуинтервале [-1/2$k$, 1/2$k$). Можно показать, что значения Y равновероятны, а Z распределена равномерно. Случайные величины Y и Z независимы, поэтому математическое ожидание и дисперсия X могут быть получены сложением соответствующих величин для Y и Z. M[Y] = M[Z] = 0, D[X] = 1/12, D[Z] = 1/12k$^2$. Дисперсию D[Y] можно вычислить как разность D[X] - D[Z] либо непосредственно.

Выборку на выходе датчика ПСЧ обозначим $\{x_i\}$, $i$=1, 2,... $n$. Величины $x_i$ моделируют значения случайной величины X. Из выборки $\{x_i\}$ с помощью (4.1) получим выборки $\{y_i\}$и $\{z_i\}$ для случайных величин Y и Z, соответственно. В отсутствие фильтрации можно принять, что выборки содержат независимые значения случайных величин X, Y и Z. Также нет зависимости между Y и Z. Это обеспечивается качеством датчика и может быть проверено эмпирически.

Фильтрация состоит в том, что из выборки $\{y_i\}$ удаляется очередной элемент, нарушающий ограничение $|n_i - n_j| \le d$, где $n_j$ - накопленное количество значений $Y_j$ в выборке, $j$ = 1, 2, ... $k$, $d$ - порог фильтрации. Соответственные элементы удаляются также из выборок $\{x_i\}$и $\{z_i\}$. В результате значения $y_i$. становятся зависимыми от предыдущих. Как показано далее, зависимость возникает и усиливается по мере роста размера выборки $n$. Поскольку фильтрация не зависит от значений $z_i$, для них пропуск значений является случайным, а оставшиеся значения по-прежнему независимы.

По свойствам датчика ПСЧ, оценки математического ожидания и дисперсии для выборок $\{x_i\}$, $\{y_i\}$ и $\{z_i\}$ должны сходиться к соответствующим значениям для случайных величин X, Y и Z, соответственно. При включении фильтрации происходит выравнивание частот $n_j$, что усиливает сходимость оценок для Y к теоретическим значениям. Фильтрация не влияет на статистические свойства выборок Z.

Влияние взаимозависимости значений $y_i$ при фильтрации можно проследить на примере оценки математического ожидания X, $\tilde{m}(\mathrm{X})$. Эта случайная величина равна среднему арифметическому $n$ одинаково распределенных случайных величин X. В отсутствие фильтрации принимается, что указанные случайные величины независимы. В этом случае дисперсия $\mathrm{D}[\tilde{m}(\mathrm{X})]$ равна дисперсии D[X] (т.е. 1/12), деленной на $n$. На Рис. 6 показано отношение оценки дисперсии $\mathrm{D}[\tilde{m}(\mathrm{X})]$ для серии из 200 выборок к теоретическому значению 1/12$n$ (кривая d=∞). Это отношение близко к единице в широком диапазоне изменения n.

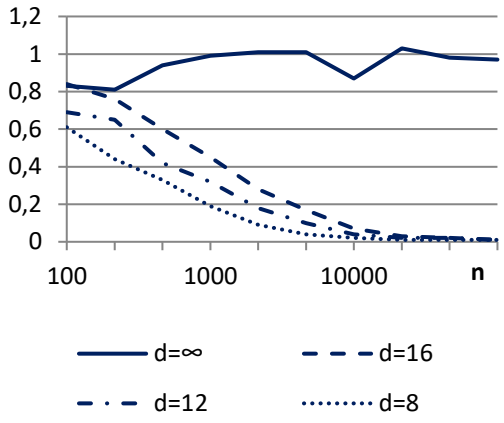



Рис. 6. Отношение оценки дисперсии $\mathrm{D}[\tilde{m}(\mathrm{X})]$ для серии выборок к значению 1/12$n$ при разных порогах фильтрации.

При фильтрации пропуск значений зависит от совокупности значений, выданных датчиком ПСЧ ранее. Тем самым, значения в выборке не будут независимыми. Как показано на Рис. 6, при этом оценка дисперсии $\mathrm{D}[\tilde{m}(\mathrm{X})]$ убывает с ростом $n$ быстрее, чем при независимости значений. Характер этого убывания можно оценить, используя представление X в виде суммы Y + Z (см. выше). Так как значения Z независимы, дисперсию $\mathrm{D}[\tilde{m}(\mathrm{Z})]$ можно считать близкой к теоретическому значению $1/12nk^2$. Вычислим также оценку $\tilde{m}(\mathrm{Y})$, дисперсию которой нужно найти:

$$\tilde{m}(\mathrm{Y}) = \frac{1}{n}\sum_{j=1}^{k} n_j Y_j =$$

$$= \frac{1}{n}\sum_{j=1}^{k}(n_j - n_{min})Y_j \qquad (4.2)$$

Здесь, как и выше, $n_j$ - число значений $Y_j$ в выборке, $n_{min}$ –минимум $n_j$ для $j$=1, 2,,...$k$. Сумма $n_{min}Y_j$ нулевая в силу четности набора $Y_j$. Из-за ограничения $0 \le n_j - n_{min} < d$ сумма в (4.2) ограничена, и оценка $\tilde{m}(\mathrm{Y})$ убывает как 1/$n$. Так как Y и Z взаимно независимы, дисперсия $\mathrm{D}[\tilde{m}(\mathrm{X})]$ равна сумме $\mathrm{D}[\tilde{m}(\mathrm{Y})]$ и $\mathrm{D}[\tilde{m}(\mathrm{Z})]$, откуда получим:

$$\mathrm{D}[\tilde{m}(\mathrm{X})] = \mathrm{M}\left[\left(\tilde{m}(\mathrm{Y})\right)^2\right] + \mathrm{D}[\tilde{m}(\mathrm{Z})] =$$

$$= \frac{C}{n^2} + \frac{1}{12nk^2}, \qquad (4.3)$$

где $C = C\,(n, d)$ – ограниченная функция, равная дисперсии сумм $n_jY_j$. Значения $C$ можно получить непосредственно из (4.3), подставляя оценки $\mathrm{D}[\tilde{m}(\mathrm{X})]$ из численных экспериментов. Эти оценки для датчика MT19937 показаны на Рис.7.

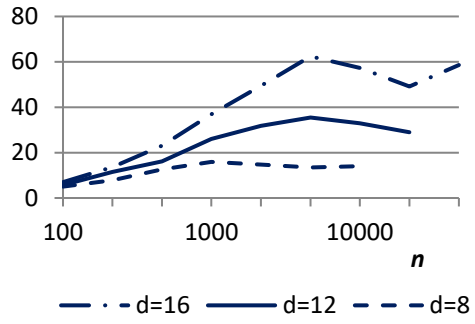



Рис. 7. $C(n, d)$, дисперсия оценки сумм дискретной случайной величины Y для разных значений порога фильтрации $d$.

На Рис. 7 видно, что $C(n, d)$ сначала растет с $n$, затем выходит на некоторый уровень, приближенно пропорциональный $d^2$. Чем больше порог фильтрации $d$, тем меньше ограничение. В отсутствие фильтрации ограничения нет, и дисперсия оценки сумм дискретной составляющей оказывается пропорциональной $1/n$, а не $1/n^2$. На этом примере можно видеть, как влияет взаимозависимость элементов выборки.

Рассмотрим также частоту выдачи пар одинаковых значений для дискретного равномерного распределения на целочисленном отрезке небольшой длины. При независимых значениях в выборке вероятность совпадения с предыдущим значением равна $1/k$, где $k$ - число возможных значений. Будем анализировать произведение полученной частоты таких пар $p$ на $k$, $T = p \cdot k$, осредненное по серии выборок. В экспериментах число выборок 200, длина выборки 4000. Результаты показывают, что в отсутствие фильтрации $T$ отличается от единицы не более чем на 1% и не зависит от длины отрезка. При $F$=16 отрезках фильтрации и для порога фильтрации $d$=12 $T$ меняется от 0.97 до 0.95 при изменении длины отрезка $n$ от 8 до 32. С ростом порога (и с ослаблением фильтрации) $T$ приближается к единице. Таким образом, при фильтрации частота пар одинаковых значений снижается на несколько процентов. Это объясняется устройством механизма фильтрации: у пары одинаковых значений вероятность превысить порог выше, чем у одиночного значения. При этом второе значение пропускается, и пары больше нет.

## 5. Заключение

Наиболее известные датчики случайных чисел могут давать недостаточно равномерное распределение для генерации случайных тестов. Повысить равномерность для любого датчика может фильтрация на основе выравнивания гистограммы выдаваемых чисел. Тем самым снижается их независимость, однако это не ухудшает статистические свойства выборок. Подбор параметров фильтрации позволяет получить нужную равномерность распределения при относительно небольшой доле пропусков.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2024-0003 "Методы разработки аппаратно-программных платформ на основе защищенных и устойчивых к сбоям систем на кристалле и сопроцессоров искусственного интеллекта и обработки сигналов".

## 6. Приложение. Схемы использованных датчиков ПСЧ

Линейный конгруэнтный генератор (LCG): рекурсивный, вырабатывает 32-битовое целое с помощью 64-битовых операций:

```
t = 214013 * x[i] + 2531011;
x[i+1] = (t ^ (t >> 15)) & 0xFFFFFFFF
```

Генератор XorShift: также рекурсивный, вырабатывает 32-битовое целое, операции 32-битовые. Первая версия:

```
s = x[i] ^ (x[i] << 13);
t = s ^ (s >> 17);
x[i+1] = t ^ (t << 5)
```

Во второй версии алгоритм использует первое и последнее из четверки ранее выработанных чисел $x_i$, $x_{i-1}$, $x_{i-2}$, $x_{i-3}$.

```
s = x[i-3] ^ (x[i-3] << 11);
t = s ^ (s >> 8);
x[i+1] = (t ^ x[i]) ^ (x[i] >> 19)
```

В этой версии для инициализации нужны четыре числа. Одно из них задавалось "зерном", три других получались умножением "зерна" на простые числа 8179, 8191, 8209.

Генератор Mersenne Twister: первая версия взята из www.agner.org/random в 2003 году. Она использует циклический буфер $z$ размером $K$=17 пар 32-битовых чисел. На каждом шаге текущая позиция $i$ в буфере обновляется и затем циклически сдвигается назад.

```
struct {int x, y;} z [K];
t = _lrotl (z[i].x, 19) + z[i+10].x;
z[i].x = _lrotl (z[i].y, 17) +
         z[i+10].y;
z[i].y = t;
i = i-1 при i>0, иначе i = K-1.
```

Функция _lrotl() выполняет побитовый цик-

лический сдвиг влево. Таким образом, на каждом шаге генерируется пара ПСЧ, но для выработки 32-битового целого используется только второе (т.е. *t*). Пара ПСЧ используется при выработке числа с плавающей точкой двойной точности.

Вторая версия генератора Mersenne имеет обозначение MT19937 и использует циклический буфер *x*, содержащий *K*=624 32-битовых числа. На каждом шаге текущая позиция *i* в буфере обновляется и затем циклически сдвигается вперед. Перед выдачей числа делается дополнительное перемешивание его битов, т.н. "закалка".

Обновление текущей позиции:

```
j=(i+1)mod K, m=(i+397)mod K

t = (x[i] & 0x80000000) |
    (x[j] & 0x7FFFFFFF);
Если t&1 == 0, t =  t >> 1;
Иначе t = (t >> 1) ^ 0x9908B0DF;
x[i] = t = t ^ x[m];
```

Сдвиг текущей позиции:

```
i = j;
```

"Закалка":

```
t = t ^ (t >> 11);
t = t ^ ((t << 7)  & 0x9D2C5680);
t = t ^ ((t << 15) & 0xEFC60000);
t = t ^ (t >> 18);
```

# Improving the Uniformity of Pseudorandom Numbers

**A.S. Koutsaev**

**Abstract.** The quality of random tests generation depends on the choice of random number generator. Testing popular generators using the Pearson chi-square test shows that a uniform distribution with a significance level of 70% or higher is observed for only half of the samples. For this reason, to obtain the necessary coverage, you need to increase the volume of tests. A method is proposed to increase the uniformity of distribution for random number generators, based on filtering the sample. Selection of filter parameters allows you to obtain the desired uniformity of distribution with a moderate number of rejects.

**Keywords:** random number generator, uniform distribution, chi-square, filtering, random tests.

## Литература

1. D. H. Lehmer, Mathematical methods in large-scale computing units, Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery, 1949, Harvard University Press, Cambridge, Mass., 1951, P. 141-146. MR 0044899 (13,495f)

2. G. Marsaglia. Xorshift RNGs. Journal of Statistical Software, 2003, Vol. 8, P. 1-6.

3. www.agner.org/random.

4. M. Matsumoto, T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation, 1998, Vol. 8 (1), P. 3-30.

5. F.O. Panneton, P. l'Ecuyer, M. Matsumoto. Improved long-period generators based on linear recurrences modulo 2. ACM Transactions on Mathematical Software, 2006, 32 (1), P. 1-16.

6. Е.С. Вентцель, Л.А. Овчаров. Теория вероятностей и ее инженерные приложения. М., "Наука", 1988.

# Динамический анализ и оптимизация ввода-вывода в среде виртуализации GNU Linux/QEMU/KVM

**А.Б. Бетелин[1], Г.А. Прилипко[2], А.Г. Прилипко[3], С.Г. Романюк[4], Д.В. Самборский[5]**

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

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, prilipko@niisi.msk.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, aleksey.prilipko@gmail.com;

[4]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, sgrom@niisi.ras.ru;

[5]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, samborsky_d@fastmail.com

**Аннотация.** В данной статье приведены результаты тестирования производительности ввода-вывода виртуальных операционных систем в среде виртуализации GNU Linux/QEMU/KVM и предложены способы увеличения производительности приложений виртуальных ОС с помощью более оптимального использования файлового кэша ОС Linux. Разработана утилита динамического анализа и оптимизации использования виртуальных дисков. Тестирование показало, что использование данной утилиты позволяет достичь более чем двукратного ускорения смешанных нагрузок ввода-вывода на SSD-накопителях.

**Ключевые слова:** виртуализация, ввод-вывод данных, QEMU, KVM, Linux, vmtouch, vmprobe, mincore

## 1. Введение

Система виртуализации QEMU/KVM [1, 2] в операционной системе GNU Linux предлагает несколько режимов подключения виртуальных или реальных устройств накопителей данных (так называемых «дисковых» устройств). Накопителем данных для виртуальной ОС может выступать обычный файл, дисковый том, SATA- или SCSI-устройство, NVMe-накопитель данных. Подключение файла или дискового тома в качестве виртуального накопителя данных осуществляется либо в режиме эмуляции некоторого протокола (SATA, SCSI), либо с помощью драйвера семейства VirtIO, использующего технологию паравиртуализации, который должен быть установлен в виртуальную ОС. В последнем случае VirtIO-драйвер напрямую указывает буфер данных и команду записи или чтения библиотеке гипервизора, которая выполняет чтение или запись без лишних накладных расходов.

Для делегирования виртуальной ОС монопольного доступа к накопителю данных используются драйверы vhost-scsi, vhost-user, vfio-pci. Такое подключение накопителей данных обеспечивает максимальную общую производительность и минимальное время выполнения отдельных команд ввода-вывода. Но при этом усложняется настройка сервера виртуализации и становятся невозможными функции моментальных бэкапов (live snapshot), безостановочной миграции (live migration), управление скоростью ввода-вывода (IO throttling) и др.

Поэтому при проектировании вычислительного кластера для среды виртуализации QEMU/KVM приходится находить компромисс между требуемой скоростью ввода-вывода и гибкостью администрирования. Обычно, если не требуется достичь 100% скорости накопителя, то для локальных накопителей оказывается оптимальным использование драйверов семейства VirtIO. Так, в статье [3] было показано, что даже при использовании быстрого накопителя Intel Optane 900P драйвер virtio-scsi с выделенным потоком управления ввода-вывода (iothread) показывает скорость записи блоков данных размером 4Кб всего на 25% меньше, чем при использовании драйвера vhost-user и библиотеки Intel SPDK для прямого доступа к устройству. В случае блоков размером 2Мб разницы в производительности между virtio-scsi и vhost-user не наблюдается.

При подключении виртуальных дисков с помощью VirtIO драйверов система QEMU использует стандартный стек ввода-вывода ядра Linux, поэтому при работе виртуального диска может быть задействован системный файловый кэш (называемый «page cache» в терминологии ядра Linux). Согласно некоторым популярным рекомендациям кэширование SSD-накопителей не ускоряет доступ к данным, и его советуют выключать. Тем не менее, любое подобное общее утверждение рекомендуется проверять тщательным тестированием. Чтобы исследовать влияние

файлового кэша на производительность виртуальных дисков, авторами были выполнены соответствующие тесты и предложены рекомендации по увеличению скорости ввода-вывода виртуальных ОС QEMU/KVM.

Для ускорения ввода-вывода с помощью более оптимального использования файлового кэша была разработана утилита vmdisktouch, выполняющая динамический анализ использования виртуального диска и своевременную загрузку часто используемых областей диска в файловый кэш. На момент публикации авторы не имеют сведений о наличии другого общедоступного инструмента с функциями, аналогичными функциям утилиты vmdisktouch.

## 2. Конфигурация среды виртуализации QEMU

В данной работе были рассмотрены следующие конфигурации серверов:

- 1 или 2 процессора семейства Intel Xeon;
- от 128Гб до 512Гб оперативной памяти;
- несколько SSD- и HDD-накопителей данных, объединенных в RAID1- или RAID10- массивы.

Эта конфигурация соответствует серверу средней ценовой категории и оптимальна по соотношению цены оборудования к производительности виртуализированных приложений, если нет необходимости в безостановочной миграции виртуальных ОС. На сервере подобной конфигурации могут работать несколько виртуальных ОС, выполняющих функции файл-серверов, серверов приложений и программных сетевых маршрутизаторов.

Драйвер локальных виртуальных дисков системы QEMU имеет параметр cache, задающий режим, в котором QEMU оперирует с файлом или устройством виртуального диска.

Четыре возможных значения этого параметра соответствуют четырем комбинациям флагов O_DIRECT и O_DSYNC системного вызова открытия файла: cache=none – O_DIRECT; cache=writethrough – O_DSYNC; cache=directsync – O_DIRECT и O_DSYNC; cache=writeback – без флагов. По стандарту POSIX [9] опция O_DIRECT запрещает кэширование данных в общесистемном файловом кэше, а опция O_DSYNC указывает на необходимость ожидания подтверждения от устройства накопителя каждой операции записи данных.

Режим cache=writeback не является надежным, так как он не гарантирует сохранности записанных данных при внезапном отключении питания. С другой стороны, полное отключение кэширования данных [1] (режимы cache=none и cache=directsync) не рекомендуется для всех накопителей данных кроме самых скоростных NVMe и NVRAM устройств, имеющих пропускные способности и задержки более близкие к характеристикам оперативной памяти, чем к дисковым устройствам. Таким образом, оптимальным режимом работы виртуальных дисков для локальных накопителей является режим кэширования данных со сквозной записью, cache=writethrough.

## 3. Измерения влияния файлового кэша на производительность ввода-вывода

Для оценки возможности ускорения дискового ввода-вывода в случае попадания данных в файловый кэш был выполнен набор тестов производительности, предоставляемых утилитой Flexible I/O (FIO) версии 3.25 [4]. Тесты выполнялись из виртуальной ОС по отношению к виртуальным дискам (файлам в формате RAW). Каждый тест выполнялся либо с виртуальным диском, полностью загруженным в файловый кэш и удерживаемым в нем командой vmtouch -l ..., либо наоборот, полностью вытесненным из кэша командой vmtouch -e ... перед запуском теста.

В таблицах 1 и 2 собраны результаты тестирования производительности виртуального диска на SSD и HDD устройствах в двух режимах:

- многопотоковый асинхронный режим, командой вида

```
fio -ioengine=libaio -rw=randrw -rwmixread=80 -bs=4k -direct=1 -iodepth=128 -numjobs=4 -sync=0 ...;
```

- однопотоковый синхронный режим, командой вида

```
fio -ioengine=libaio -rw=randrw -rwmixread=80 -bs=4k -direct=1 -iodepth=1 -numjobs=1 -sync=1 ....
```

Опции -rwmixread=80 -bs=4k задают режим тестирования чтения и записи блоков данных размеров 4096 байта в отношении 4 к 1 (80% — чтение, 20% — запись). В тестах SSD устройств (таблица 1) были задействованы два NVMe-накопителя данных модели Intel SSD Pro 6100p, объединенных в программный массив RAID1. В тестах HDD устройств (таблица 2) применялись четыре HDD-накопителя данных модели Seagate ES.3 ST2000NM0033, объединенных в программный массив RAID10. В обоих случаях виртуальным диском служил файл в формате RAW,

который был создан в режиме полной инициализации.

Тестовая утилита FIO запускалась из виртуальной ОС AlmaLinux 9. Для работы с виртуальным диском использовался драйвер диска VirtIO, наиболее производительный из драйверов виртуальных дисков в системе QEMU. Применение данного драйвера не добавляет накладных расходов эмуляции протокола обмена по шинам SCSI или SATA.

Основной результат тестов, приведенных в таблице 1, состоит в том, что использование файлового кэша позволяет ускорить в 2.36 раза скорость ввода-вывода SSD-накопителей в однопотоковом синхронном режиме. Из данных таблицы следует, что ускоряются не только операции чтения, что очевидно при кэшировании, но и операции записи. Приведенные в таблице значения задержки операций (последние три строки) означают полное время выполнения операции от ее запуска до получения подтверждения об окончании. Поскольку тесты FIO выполнялись с опцией -sync=1, которая форсировала вызов синхронизации данных после каждой записи, то получение подтверждения означало, что данные действительно записаны в постоянную память SSD-устройства. Из последней строки видно, что среднее время записи уменьшилось почти в 2 раза, тогда как минимальное время записи почти не изменилось. Это позволяет предположить, что при интенсивном использовании файлового кэша в очередь запросов SSD-устройства попадают преимущественно операции записи, и в этом режиме устройство реже останавливается для выполнения внутренних операций, таких, как сборка мусора и планирование равномерного использования ресурса перезаписи ячеек памяти.

Таблица 1. Результаты тестирования смешанной нагрузки ввода-вывода на SSD-накопителях

| Тест | Кэш пуст | Кэш предзагружен | Ускорение |
|---|---|---|---|
| Многопотоковый асинхронный режим | | | |
| Чтение, операций в секунду | 65200 | 73700 | 1.13 |
| Запись, операций в секунду | 16300 | 18400 | 1.13 |
| Всего, операций в секунду | 81500 | 92100 | 1.13 |
| Минимальная задержка записи, µсек | 2040 | 2080 | 1.02 |
| Медианная задержка записи, µсек | 16710 | 14350 | 1.16 |
| Средняя задержка записи, µсек | 17975 | 15760 | 1.14 |
| Однопотоковый синхронный режим | | | |
| Чтение, операций в секунду | 2090 | 4900 | 2.34 |
| Запись, операций в секунду | 510 | 1240 | 2.43 |
| Всего, операций в секунду | 2600 | 6140 | 2.36 |
| Минимальная задержка записи, µсек | 348 | 346 | 1.01 |
| Медианная задержка записи, µсек | 502 | 470 | 1.07 |
| Средняя задержка записи, µсек | 1010 | 535 | 1.89 |

Данный эффект противоречит популярным рекомендациям, согласно которым использовать файловый кэш по отношению к SSD-накопителям не имеет смысла и снижает производительность ввода-вывода. Опровержение этих рекомендаций открывает различные возможности оптимизации подсистемы ввода-вывода. Например, заметного ускорения можно ожидать при добавлении высокоскоростного NVMe- или PCIe-накопителя небольшого объема в качестве кэша системы логических томов LVM (основанного на модуле ядра dm-cache [5]). Данный кэш будет работать в режиме сквозной записи, что не снизит надежность всей системы, но при этом ускорится работа виртуальных дисков, расположенных на этом томе.

В случае многопотоковой асинхронной нагрузки наблюдаемое ускорение операций составляет всего 13%. В этих тестах SSD-устройство показывает значительно большую производительность, чем в однопотоковом тесте с синхронной записью. Но среднее время выполнения операции записи в этом тесте тоже выше более чем 15 раз, нежели в синхронном режиме. Асинхронная многопотоковая нагрузка — это тот тип нагрузки, для которого оптимизирована внутренняя программа SSD-устройства, поэтому в таком режиме устройство показывает наибольшую производительность. Кроме того, поскольку тестируемая модель SSD-накопителей, согласно ее спецификации, оптимизирована для операций чтения, то основное время в этом тесте расходовалось на запись данных, и использование файлового кэша не дало заметного прироста производительности.

Таблица 2. Результаты тестирования смешанной нагрузки ввода-вывода на HDD-накопителях

| Тест | Кэш пуст | Кэш предзагружен | Ускорение |
|---|---|---|---|
| Многопотоковый асинхронный режим | | | |
| Чтение, операций в секунду | 450 | 852 | 1.89 |
| Запись, операций в секунду | 113 | 211 | 1.87 |
| Всего, операций в секунду | 563 | 1063 | 1.89 |
| Минимальная задержка записи, µсек | 261940 | 48200 | 5.43 |
| Медианная задержка записи, µсек | 1166015 | 677330 | 1.72 |
| Средняя задержка записи, µсек | 1145420 | 624950 | 1.83 |
| Однопотоковый синхронный режим | | | |
| Чтение, операций в секунду | 53 | 145 | 2.74 |
| Запись, операций в секунду | 13 | 35 | 2.69 |
| Всего, операций в секунду | 66 | 180 | 2.73 |
| Минимальная задержка записи, µсек | 6745 | 4470 | 1.51 |
| Медианная задержка записи, µсек | 37770 | 21365 | 1.77 |
| Средняя задержка записи, µсек | 40108 | 26930 | 1.52 |

Из таблицы 2 видно, что для HDD-устройств наблюдается 2.7x и 1.9x кратное ускорение для синхронной и асинхронной работы, соответственно. Но причины ускорения от использования кэширования в случае HDD отличаются от случая SSD. Скорость доступа к данным у механических дисковых накопителей значительно ниже, чем у твердотельных SSD-накопителей и в основном определяется скоростью точного позиционирования блока магнитных головок к дорожке с данными. Следовательно, операции чтения или записи произвольного блока данных в механическом дисковом накопителе должны иметь одинаковые средние времена выполнения (если не рассматривать устройства с «черепичной» записью — Shingled Magnetic Recording, SMR), тогда как запись данных SSD-устройством выполняется медленнее чтения. Также можно считать, что в тестах с пустым кэшем все операции чтения выполняются непосредственно из накопителя (так как диск большой и случайное чтение за время теста имеет незначительное количество повторно прочтенных блоков), а в тестах с полностью загруженным кэшем все операции чтения выполняются из кэша. Поэтому в асинхронном многопотоковом режиме можно было бы ожидать 5-кратного увеличения количества операций записи, так как все операции чтения выполняются из кэша, а их в смешанной нагрузке 80%. Но наблюдаемое увеличение меньше 5-кратного — от 113 до 211 операций в секунду. Это расхождение частично объясняется тем, что в массиве RAID10 операции чтения могут выполняться параллельно, а операции записи — нет. Если учесть эту поправку, то увеличение скорости записи должно быть не 5-кратным, а 3-кратным. Но поскольку 211 операций записи в секунду приблизительно соответствуют оценке пиковой производительности произвольного доступа для HDD-накопителя, вопрос скорее в том, почему в первой колонке (тест с пустым кэшем) удалось выполнить 563 операции в секунду. Вероятно, в режиме смешанной асинхронной нагрузки жесткий диск имеет больше возможностей по оптимизации плана выполнения запросов и более успешно использует внутреннюю кэш-память. В тесте синхронного доступа небольшое количество операций записи (35 операций/сек) совпадало с результатами аналогичного теста на сервере виртуализации. HDD-накопители со скоростью вращения 7200 об/мин выполняют одиночную операцию записи за среднее время 10-15мсек, что соответствует 67-100 операциям/сек. Но в режиме RAID10 требуется ожидать подтверждения записи двух копий данных от двух устройств, что увеличивает среднее время операции. Кроме того, тестируемые HDD-накопители находились в эксплуатации более 8 лет, и их высокий износ мог привести к увеличению времени точного позиционирования блока головок.

# 4. Программа динамической оптимизации кэширования виртуальных дисков

Приведенные выше результаты тестов демонстрируют возможность ускорения ввода-вывода за счет применения файлового кэша для виртуальных дисков в случае своевременной загрузки необходимых данных.

Кэширование данных открытых файлов выполняется в рамках работы более общего механизма управления использованием страниц памяти. Традиционно ядро Linux создает два списка страниц памяти для алгоритма удержа-

ния в ОЗУ наиболее часто используемых страниц (least-recently-used, LRU). Страницы, к которым недавно осуществлялся доступ, помещаются в начало списка «активных» страниц. Страницы из хвоста этого списка удаляются, если к ним давно не обращались, и помещаются в начало списка «неактивных» страниц. Когда какой-либо процесс повторно обращается к «неактивной» странице, она помещается обратно в список «активных» страниц. В случае дефицита памяти ядро удаляет первыми страницы из хвоста списка «неактивных» страниц. Если список «неактивных» страниц становится короче половины длины списка «активных» страниц, то выполняется перераспределение страниц этих списков для сохранения указанного соотношения длин списков. Следует отметить, что для определения факта использования страниц памяти применяется бит Accessed в структуре Page Table Entry (PTE), который автоматически выставляется устройством управления памятью (MMU).

Таким образом, алгоритм LRU приближенно решает задачу оценки вероятности будущего доступа к ранее использованной странице памяти, почти не задействуя для этого вычислительные ресурсы. В современных версиях ядра Linux картина усложняется, если имеется несколько процессоров с неоднородной архитектурой памяти (non-uniform-memory-access, NUMA) или применяется механизм изоляции ресурсов cgroups — тогда вышеописанные списки создаются для каждого процессора и каждой группы процессов cgroups. Подробное описание механизма управления памятью, список соответствующих ему параметров и методов диагностики содержится в документации ядра Linux [6].

Отметим следующие недостатки общесистемного файлового кэша:

- он универсален — у пользователя нет возможности выделить больше памяти для определенных файлов;
- он имеет небольшую гранулярность, равную размеру страницы памяти (4096 байт), при этом нет возможности задать больший размер блоков, чтобы данные загружались «по ассоциации».

Частично решить задачу распределения приоритетов в использовании оперативной памяти позволяет современный механизм изоляции ресурсов cgroups v2, имеющий параметры memory.low и memory.high для каждой группы процессов. Эти параметры влияют на принятие решений об увеличении или уменьшении количества страниц памяти, необходимых данной группе процессов. В среде виртуализации Libvirt/QEMU/KVM каждая виртуальная ОС помещается в отдельную группу cgroup, поэтому такой способ оптимизации вполне возможен. Однако, этот метод требует точной оценки потребности в оперативной памяти всех виртуальных ОС на сервере и не гарантирует успеха, поскольку указываются рекомендуемые размеры всей памяти для cgroup, что не обязательно приведет к более активному кэшированию содержимого виртуального диска, скорость работы которого требуется увеличить.

Еще одним параметром, косвенно влияющим на работу файлового кэша, является размер окна опережающего чтения (read-ahead size), который можно задать либо для блочного устройства на сервере виртуализации, либо внутри виртуальной ОС. Но увеличение этого параметра не всегда приводит к желаемому результату, так как оно повышает количество прочитываемых и удерживаемых в кэше данных, не все из которых оказываются нужны.

С другой стороны, задача приоритетного удержания часто используемых данных виртуальных дисков допускает и более простое решение. Так, ядро Linux имеет системный вызов mincore, который возвращает список находящихся в файловом кэше блоков указанного файла. К этому системному вызову обращается утилита vmtouch [7], которая в зависимости от указанных параметров либо сообщает пользователю, какая часть интересующего его файла находится в кэше, либо загружает и удерживает весь этот файл или некоторую его часть в кэше. Попытки применить утилиту vmtouch для удержания в кэше всего виртуального диска или некоторой его части обычно не реалистичны, так как часть данных диска, которая будет интенсивно использоваться, заранее неизвестна, а размер всего диска, как правило, превосходит размер оперативной памяти сервера. Исключением служит ситуация, когда для ускорения работы некоторого приложения в виртуальной ОС создается отдельный виртуальный диск небольшого размера, а затем файл этого диска полностью удерживается в кэш-памяти утилитой vmtouch. Развитием утилиты vmtouch является утилита vmprobe [8], которая также имеет функции сохранения «моментального снимка» состояния кэш-памяти применительно к указанному файлу и последующего восстановления состояний кэш-памяти из сохраненных таким образом «снимков». Эти функции утилиты vmprobe позволяют, например, зафиксировать определенное состояние сервера базы данных, часто называемое «прогретым» состоянием, и в дальнейшем при старте сервера быстрее достигать этого состояния, форсируя загрузку необходимых данных в файловый кэш. Тем не менее, утилиты vmtouch и vmprobe не проводят динамического

анализа использования файлового кэша, и поэтому не способны перенастраивать множество загруженных страниц данных в реальном времени.

Для более гибкого управления кэшированием виртуальных дисков авторами данной статьи была разработана утилита vmdisktouch, которая является развитием утилиты vmtouch и выполняет следующие действия:

- периодически делает системный вызов mincore и составляет карту загруженных в файловый кэш блоков виртуального диска;
- находит области диска, где недавно выполнялось много операций чтения и записи, и помечает их как области, которые имеет смысл удерживать в кэше;
- удерживает в кэше данные выбранных областей виртуального диска;
- выводит «тепловую карту» файлового кэша с обозначением всех удерживаемых на данный момент в кэше блоков (в том числе недавно перезаписанных) и областей.

Утилита написана на языке Python с использованием библиотек NumPy и SciPy для обработки массивов данных, библиотек mmap и fincore для работы в виртуальной памятью системы Linux, и библиотеки Pillow для вывода графических PNG файлов. Функции данной утилиты дают администратору сервера виртуализации возможность наблюдать за использованием файлового кэша по отношению к виртуальным дискам и выбирать оптимальный режим динамической загрузки часто используемых областей виртуального каждого диска.

## 5. Алгоритм работы утилиты vmdisktouch

Чтобы определить часто используемые области виртуального диска, утилита vmdisktouch применяет к битовой карте блоков файлового кэша операцию сглаживания (фильтр Гаусса) с заданной дисперсией и амплитудой. Получившаяся функция обновляет более медленно меняющуюся функцию, в которой накапливается экспоненциально затухающее среднее значение. Последнюю функцию можно интерпретировать как функцию интенсивности доступа или «тепловую карту» использования виртуального диска. Она представляет собой приближение к функции плотности вероятности событий ввода-вывода в массиве блоков данных виртуального диска. Далее в массиве этой функции выделяются сегменты блоков данных, в которых значения превышают некоторый порог. Эти сегменты затем становятся областями, которые будут удерживаться в файловом кэше.

Поскольку в результате работы этого алгоритма все блоки найденных областей попадают в кэш, их биты принудительно обнуляются перед применением операции сглаживания. Это выключает положительную обратную связь, которая иначе привела бы к монотонному распространению этих областей на все пространство блоков диска. Кроме того, обнуление этих битов приводит к постепенному уменьшению функции в данном регионе, поэтому удерживаемая в кэше область данных через некоторое время перестает удерживаться, но только если в этой области не было большого числа операций записи данных, которые также увеличивают значения функции (см. ниже). Такое динамическое переопределение множества областей, удерживаемых в кэш-памяти, помогает находить регионы диска, в которых наблюдается высокая интенсивность операций ввода-вывода.

Для обнаружения операций записи, выполняемых виртуальной ОС, утилита периодически прочитывает и вычисляет контрольные суммы блоков данных удерживаемых в файловом кэше областей. Перезапись данных виртуального диска сопровождается обновлением содержимого файлового кэша, поэтому утилита обнаружит измененные контрольные суммы и пометит блоки как перезаписанные. Утилита не может обнаруживать изменение в остальных блоках данных, поскольку само событие их чтения повлияло бы на работу файлового кэша и заставило продолжать хранить эти блоки в кэше, часто безосновательно.

Согласно логике данного алгоритма, утилита в начале своей работы наблюдает за картой загруженных в кэш блоков данных и находит области с высокой плотностью таких блоков. Затем наиболее значимые области выбираются для их полной загрузки в файловый кэш. Далее в процессе работы поступает информация о перезаписываемых блоках и добавляет информацию об областях, где данные не только читаются, но и перезаписываются. Эти области получают наибольший приоритет и дольше удерживаются в файловом кэше.

Утилита vmdisktouch принимает следующие параметры:

- имя файла виртуального диска;
- максимальный размер данных для загрузки в файловый кэш;
- минимальное значение свободной общесистемной памяти, при котором утилите разрешено загружать в кэш страницы памяти;
- длительность цикла работы утилиты;
- параметры амплитуды и ширины Гауссова фильтра для карты блоков файлового кэша и отдельно для карты перезаписанных блоков;

• коэффициент затухания для экспоненциального среднего значения функции интенсивности доступа;

• порог значения функции интенсивности доступа для определения областей, загружаемых в файловый кэш;

• имена PNG-файлов, в которые выводятся два изображения: «тепловой карты» для функции интенсивности и изображение, где точно обозначены блоки данных, загруженные в файловый кэш;

• параметры вывода изображения: число блоков на пиксел, соотношение сторон, и масштаб изображения.

Обязательными параметрами являются только первые два, для остальных же утилита выбирает значения, близкие к оптимальным и выбранные на основании опыта ее применения. При обновлении множества удерживаемых в кэше областей виртуального диска утилита использует приоритетную очередь, в которой значение приоритета — сумма функции интенсивности доступа. Это обеспечивает оптимальность выбора таких областей, если загрузить все найденные области не позволяет ограничение на общий размер (второй параметр утилиты).

Настройка остальных параметров алгоритма утилиты устанавливает режим работы утилиты. Например, с помощью амплитуды фильтра для перезаписанных блоков можно задать более высокий приоритет тем областям данных виртуального диска, где часто выполняются операции записи. Коэффициент затухания функции и длительность цикла определяют скорость перенастройки загруженных в кэш областей данных.

За работой утилиты удобно наблюдать, открыв PNG-файл изображения «тепловой карты» виртуального диска. Многие программы просмотра изображений в системе Linux отслеживают изменение содержимого файла и обновляют картинку, что дает эффект отображения в режиме реального времени. Для отрисовки «тепловой карты» используется цветовая палитра, обозначающая тип блоков данных: синий цвет соответствует блокам, загруженным по инициативе утилиты vmdisktouch, зеленый цвет означает загрузку по инициативе виртуальной ОС, красным цветом отмечены недавно перезаписанные блоки данных.

## 6. Заключение

Разработанная авторами утилита vmdisktouch позволяет более интенсивно кэшировать данные виртуального диска и выбирать для этого преимущественно те области диска, которые участвуют одновременно в чтении и записи данных. Оказывается, что такое кэширование увеличивает также скорость операций записи SSD-накопителей, особенно в синхронном режиме. Использование данной утилиты не нарушает функционирование виртуальных ОС, поскольку ее работа заключается только в опросе состояния файлового кэша системы Linux и чтении данных виртуальных дисков.

Недостатком алгоритма работы утилиты vmdisktouch является то, что одно лишь наблюдение за блоками виртуального диска, находящимися в файловом кэше, делает затруднительным определение причины попадания этих блоков в кэш. Алгоритм должен запоминать блоки, загрузка которых была осуществлена по его инициативе, и исключать их из дальнейшего динамического анализа, иначе области распространятся постепенно на весь виртуальный диск. При этом теряется информация о том, понадобились ли некоторые из этих блоков, т.е. производились ли операции чтения этих блоков данных со стороны виртуальной ОС. По этой же причине не удается детектировать события записи блоков вне областей, уже удерживаемых в памяти. Если бы система виртуализации позволяла наблюдать за событиями и чтения и записи блоков данных виртуального диска, то можно было бы более точно определять области данных, рекомендованные для удержания в кэше.

Дальнейшим развитием функциональности утилиты vmdisktouch может быть реализация следующих возможностей:

• отслеживание перезаписанных блоков данных с помощью карты обновленных блоков виртуального диска (функция block-dirty-bitmap, добавленная в QEMU v.2.4). Этот метод позволит получать список всех выполненных операций записи данных, что увеличит точность работы утилиты и сэкономит процессорный ресурс;

• временной анализ событий ввода-вывода с обнаружением закономерностей и предсказанием будущих событий для опережающей загрузки блоков данных в файловый кэш.

Работа выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2024-0001 «Создание и реализация доверенных систем искусственного интеллекта, основанных на новых математических и алгоритмических методах, моделях быстрых вычислений, реализуемых на отечественных вычислительных системах» (1023032100070-3-1.2.1).

# A Dynamic Analysis and Optimization of I/O in the GNU Linux/QEMU/KVM Virtualization Environment

**A.B. Betelin, G.A. Prilipko, A.G. Prilipko, S.G. Romanyuk, D.V. Samborskiy**

**Abstract.** In this paper we describe results of virtual disk I/O performance tests in GNU Linux/QEMU/KVM environment and propose optimization methods that improve use of the system page cache. The authors have developed a utility for dynamic analysis and optimization of virtual disk usage. Testing has shown that this utility can provide more than 2x acceleration for mixed I/O workloads on SSDs.

**Keywords:** virtualization, I/O, QEMU, KVM, Linux, vmtouch, vmprobe, mincore

## Литература

1. Сайт "QEMU, the FAST! processor emulator". https://www.qemu.org (дата обращения 06.03.2024).

2. Сайт проекта KVM. https://www.linux-kvm.org (дата обращения 06.03.2024).

3. А.Б. Бетелин, И.Б. Егорычев, А.А. Прилипко, Г.А. Прилипко, С.Г. Романюк, Д.В. Самборский. Настройка и оптимизация системы ввода-вывода в среде виртуализации GNU Linux/QEMU/KVM/Libvirt. «Труды НИИСИ РАН», т.9 (2019), № 5, 119–129.

4. Сайт документации утилиты Flexible I/O tester (FIO). https://fio.readthedocs.io/en/latest (дата обращения 06.03.2024).

5. Сайт документации ядра Linux, раздел "Device Mapper. Cache". https://www.kernel.org/doc/Documentation/device-mapper/cache.txt (дата обращения 06.03.2024).

6. Сайт документации ядра Linux, раздел "Memory Management". https://www.kernel.org/doc/html/latest/admin-guide/mm (дата обращения 06.03.2024).

7. Сайт утилиты vmtouch. https://hoytech.com/vmtouch (дата обращения 06.03.2024).

8. Сайт утилиты vmprobe. https://vmprobe.com/intro (дата обращения 06.03.2024).

9. Стандарт POSIX.1-2017. The Open Group Base Specifications Issue 7, 2018 edition IEEE Std 1003.1-2017.

# Познание и использование свойств природы когнитивным автономным агентом

**В.Г. Редько**

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, vgredko@gmail.com

**Аннотация.** Построена и проанализирована модель когнитивного автономного агента, познающего простые свойства природы. Предполагается, что агент развивает способность примитивного мышления, в частности, агент обладает способностью предсказывать будущие события во внешнем мире и имеет элементарное чувство причинности. Агент формирует понятия, характеризующие внешний мир, наблюдает и анализирует явления внешнего мира. Построена и проанализирована иллюстративная компьютерная модель когнитивного автономного агента, использующего свои способности при выполнении определённой полезной работы.

**Ключевые слова:** когнитивный автономный агент, простые формы мышления, познание свойств природы, научное познание

## 1. Введение

В работе [1] анализируются перспективы направления исследований «Моделирование когнитивной эволюции». В [1] представлены проблемы, демонстрирующие актуальность этого направления исследований, характеризуются заделы моделирования когнитивной эволюции (как со стороны компьютерных наук и моделей автономных агентов, так и со стороны биологических исследований когнитивных способностей животных). Подчёркивается, что наиболее перспективный путь к изучению когнитивной эволюции состоит в построении и исследовании моделей когнитивных автономных агентов, в изучении того, как в процессе когнитивной эволюции возникло наше мышление, применимое в научном познании. В работе [1] также представлены контуры программы будущих исследований когнитивной эволюции и отдельные начальные модели этих исследований.

В настоящей работе строится и анализируется модель познания простых свойств природы когнитивным автономным агентом. Рассматривается агент, который уже обладает познавательными навыками, но эти навыки ещё довольно простые. В историческом аспекте это соответствует переходу от примитивного, первобытного мышления [2, 3] к начальным формам познания закономерностей природы. Этот переход хорошо охарактеризован в книге В.Ф. Турчина «Феномен науки» [4, гл. 8, 9]. В частности, Турчин анализирует процессы эволюционного развития мышления от первобытного к современному. Будем рассматривать агента, который ещё не обладает серьёзными математическими способностями, но уже может считать, наблюдать события во внешнем мире и пытаться анализировать закономерности в природе. Способности агента ещё далеки от полноценного научного познания, но уже формируются начальные шаги к такому познанию.

## 2. Свойства когнитивного агента

Будем считать, что агент может наблюдать события в природе, определять числа, записывать числа в таблицу. Также агент может выполнять простые арифметические действия. Для определённости будем рассматривать агента, наблюдающего и анализирующего явления погоды. Также будем считать, что агенту предоставлены простые приборы, которые измеряют время, температуру воздуха, интенсивность света в окружающей среде. Агент может пользоваться этими приборами. Будем анализировать, как агент может обнаружить простые закономерности развития погоды.

Считаем, что агент обладает свойством прогнозирования будущих событий во внешней среде. А именно, если агент наблюдает, что за событием *А* многократно следует событие *Б*, то при новом появлении события *А* агент прогнозирует, что за этим событием с большой вероятностью последует и событие *Б*. Во многих случаях это можно рассматривать как связь между причиной *А* и следствием *Б*. Такое свойство прогнозирования очень близко к чувству причинности в понимании Дэвида Юма [5]. В работе [6] была построена и проанализирована компьютерная модель агентов с таким свойством прогнозирования; эти агенты могут заранее предвидеть появление благоприятных и неблагоприятных событий в окружающей среде. Благодаря предвидению эти агенты могут заранее подготовиться к новым событиям. Также было показано, что в процессе эволюции популяции агентов агенты

со свойством прогнозирования могут преимущественно вытеснять агентов без свойства прогнозирования из популяции.

Рассматриваем автономных агентов, которые в основном анализируют события в природе самостоятельно, только используя предоставляемые им приборы для измерения характеристик событий. Поведение агентов может быть промоделировано в компьютерной программе.

## 3. Познание агентом общей динамики климата в течение года

Сначала рассмотрим, как агент запоминает продолжительность дня и ночи в течение суток. Считаем, что агент наблюдает рассвет и закат (т.е. наступление светлого и тёмного времени суток), и, используя часы, каждый день регистрирует и записывает время рассвета и заката в таблицу. Конечно, при простом наблюдении эти моменты могут определяться приближённо, но для начального обнаружения простых закономерностей точность не очень существенна. Также он ежедневно определяет продолжительность дня и ночи. Для определённости будем считать, что агент начинает наблюдения с момента начала года и ведет наблюдения в течение года. После проведения этих наблюдений он может построить графики зависимости продолжительности дня и ночи от номера дня в течение года.

Так как агент может выполнять арифметические действия, то он может проводить усреднения измеренных им величин.

Считаем, что кроме определения этого, агент измеряет температуру окружающей среды и определяет среднюю температуру в течение дня и ночи, а также среднюю температуру в течение суток. Делает он это каждый день и записывает эти данные в таблицу.

Для определённости считаем, что агент географически находится на одном месте на умеренной широте (например, в Москве). Агент выполняет все эти наблюдения в течение ряда лет, а затем проводит усреднение данных по таблицам. Используя усредненные данные, агент выделяет времена года: зиму, с низкой температурой окружающей среды, лето с высокой температурой, весну и осень с растущей и уменьшающейся температурой.

## 4. Наблюдение и предсказание отдельных явлений природы

Считаем, что агент может наблюдать и регистрировать отдельные явления природы, характеризуя их как качественно, так и количественно. Так агент наблюдает солнце и регистрирует, высоко или низко находится солнце. Агент связывает светлое и тёмное время суток с наличием и отсутствием наблюдаемого солнца на небе. Также его наблюдения показывают, что летом солнце находится на небе выше, чем зимой. Аналогично агент наблюдает и регистрирует, что солнечный свет нагревает освещаемые предметы. Более того, наблюдая многократно, как солнце освещает камень и после этого камень нагревается, агент делает вывод, что солнечный свет есть причина нагревания камня.

Наблюдая на небе тучи и их движение, агент приходит к выводу, что существует ветер, который приводит к движению туч. Достаточно большие тучи приводят к осадкам: к снегу или к дождю при низкой или высокой температуре окружающей среды.

Таким образом, агент формирует определённые знания о погоде окружающей среды, может с определённой вероятностью предсказывать характер будущих событий, характер будущей погоды. Считаем, что агент может перемещаться в рассматриваемом мире и ему могут быть предоставлены средства для приспособления к окружающему его миру, например, укрытия от дождя, помещения для обогрева при морозах. Несложно задать в рассматриваемой модели индивидуальную приспособленность агента, например, считая агента подобным подвижному роботу, для которого важны температура и влажность непосредственно того места, где агент находится. Предсказывая события в окружающей среде, агент может организовать своё поведение так, чтобы стремиться повысить свою приспособленность.

## 5. Формирование понимания явлений природы

Считаем, что агент способен наблюдать и анализировать явления природы, в том числе, анализировать причинные связи между событиями в природе. Его анализ довольно простой. Приведём примеры явлений, которые наблюдает и анализирует агент.

Считаем, что агент наблюдает за растениями и животными в течение года: наблюдает, как весной пробивается и растет трава, как начинает зеленеть листва на деревьях. Агент наблюдает рост растений. Например, он видит, что при наличии влаги из зерна пшеницы появляется росток. А если посадить этот росток в достаточно влажную почву, то маленький росток превращается в стебель. Затем стебель растет, происходит формирование колоса, далее происходит цветение, формирование зерен, сначала зеленых, а затем происходит формирование спелых зерен.

Причём если спелые зерна не убрать вовремя, то эти зерна осыпаются.

Также агент может наблюдать конкуренцию между растениями, сорняки могут задавить рост полезного растения. Вредные насекомые могут повредить нужное растение и плоды растения.

Агент может наблюдать поведение животных, в частности, домашних животных человека, питание и рост животных.

Таким образом, агент может наблюдать, анализировать и понимать достаточно простые явления природы, в том числе, агент может формировать понимание причинных связей между отдельными явлениями. Например, причиной того, что полезное растение не вырастет, может быть обилие сорняков.

## 6. Иллюстративный пример познания и использования закономерностей природы

Мысленно переносим агента с рассмотренными выше свойствами в наши дни и проанализируем его свойства на достаточно простом примере.

Рассматриваем агента, который служит помощником людей. Люди предоставляют агенту допуск в Интернет, агент может пользоваться календарём, ему доступен прогноз погоды на ближайшие дни. Будем считать, что агент контролирует технологический процесс выращивания сельскохозяйственных культур на определённой территории, служит простым агрономом-координатором отдельного участка сельскохозяйственной фирмы. Для определённости рассматриваем участок для выращивания яровой пшеницы. Используя Интернет, агент может узнать рекомендации по выращиванию пшеницы, освоить эффективные методы процессов выращивания пшеницы. То есть, агент использует опыт людей по изучению рассматриваемых процессов. Агент даёт команды начала выполнения отдельных этапов работ по выращиванию пшеницы: вспашки, боронования, посева, опрыскивания, сбора урожая. При принятии решений агент анализирует связи между рассматриваемыми им событиями. Он может анализировать причины возникновения событий. Например, агент может учитывать, что для формирования успешных всходов пшеницы при посеве почва должна быть достаточно влажной. Также надо не пропустить время сбора урожая, чтобы колосья созревшей пшеницы не успели осыпаться. В конце года подводится итог такой координационной работы агента, его эффективность оценивается по количеству собранной пшеницы.

Рассмотрим конкретную компьютерную модель такого агента-координатора. Считаем, что процесс выращивания пшеницы содержит $N$ этапов. Перед началом процесса выращивания агенту предоставлены ориентировочные значения времен начала этапов $t_{10}$, $t_{20}$, … , $t_{N0}$. Имеются оптимальные времена $t_{1OPT}$, $t_{2OPT}$, … , $t_{NOPT}$ для рассматриваемого участка и для рассматриваемого года. Агенту эти оптимальные времена не известны, но он стремится наилучшим образом выбрать эти времена с учётом погоды, наблюдений за процессом выращивания пшеницы на своём участке и известных из Интернета общих рекомендаций.

В нашей иллюстративной модели рассматриваем большое количество идентичных участков, на каждом участке имеется свой агент-консультант. Оптимизация функционирования агентов осуществляется эволюционным путём. Число участков и число агентов равно $n$. Оптимизация происходит в течение ряда поколений эволюции.

Рассмотрим, как именно отдельный агент выбирает времена этапов в нашей модели. В первом поколении агент отталкивается от предоставленных ему ориентировочных значений начала этапов $t_{10}$, $t_{20}$, ... , $t_{N0}$ и немного варьирует их с учётом погоды, наблюдений за процессом выращивания пшеницы на своём участке и известных из Интернета общих рекомендаций, т.е. прибавляет к опорным временам сравнительно небольшие добавки (положительные или отрицательные). В иллюстративной модели мы считаем, что эти добавки случайны и равномерно распределены в интервале $[-d, +d]$, где $d$ – параметр варьирования ($d > 0$, для простоты модели предполагаем, что этот параметр одинаков для всех этапов). А в конце поколения агент получает новые значения времени начала этапов: $\mathbf{t}_k(1) = \{t_{1k}(1), t_{2k}(1), \dots , t_{Nk}(1)\}$, $k$ – номер агента в популяции, $k = 1, 2, \dots, n$. Для всех агентов определяется евклидово расстояние между $\mathbf{t}_k(1)$ и оптимальными временами $\mathbf{t}_{OPT} = \{t_{1OPT}, t_{2OPT}, \dots , t_{NOPT}\}$:

$$\rho_k(\mathbf{t}_k(1), \mathbf{t}_{OPT}) = \sqrt{(t_{1k}(1) - t_{1OPT})^2 + (t_{2k}(1) - t_{2OPT})^2 + ... + (t_{Nk}(1) - t_{NOPT})^2} .$$

Далее выбирается «наилучший» агент первого поколения, т.е. тот агент, для которого расстояние $\rho_k(\mathbf{t}_k(1), \mathbf{t}_{OPT})$ минимально. Номер этого агента равен $k\_min(1)$. Во втором поколении для всех агентов начальные значения времен этапов равны начальным временам этапов этого

наилучшего агента $\mathbf{t}_{k\_min}(1)$. Далее эти времена варьируются также, как в и первом поколении эволюции. И снова определяется наилучший агент второго поколения с номером $k\_min(2)$ и соответствующие времена $\mathbf{t}_{k\_min}(2)$. Эта эволюционная процедура повторяется в течение ряда поколений, в результате чего начальные времена этапов приближаются к $\mathbf{t}_{OPT}$.

В рамках иллюстративной модели было проведено компьютерное моделирования. Использовались следующие параметры моделирования.

Оптимальные времена вспашки, боронования, посева, опрыскивания и уборки урожая $\mathbf{t}_{OPT}$ полагались равными 100, 105, 110, 160 и 230 дней с начала года (считалось, что $N = 5$). Ориентировочные значения начала этапов (которые предоставлялись агентам в самом начале эволюции) $t_{10}, t_{20}, \ldots, t_{N0}$ были равны 104, 108, 112, 150, 240, соответственно. Параметр варьирования начала этапов $d$ был переменным. Численность популяции агентов составляла 10000. На рис. 1 приведены зависимости расстояния $\rho_{k\_min}$ $(\mathbf{t}_{k\_min}(m), \mathbf{t}_{OPT})$ до оптимума для наилучшего агента от номера поколения $G$ для значений параметра варьирования $d = 1$, 3 и 10 дней.

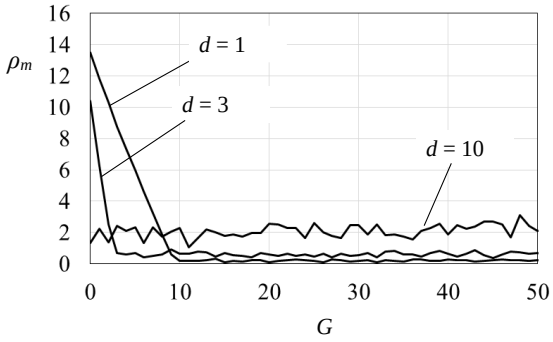



Рис. 1. Зависимости расстояния $\rho_m = \rho_{k\_min}(\mathbf{t}_{k\_min}(G), \mathbf{t}_{OPT})$ до оптимума для наилучшего агента от номера поколения $G$ для $d = 1, 3, 10$

Рис. 1 показывает, что при большом значении параметра $d = 10$ практически не происходит приближения к оптимуму. При $d = 1, 3$ приближение к оптимуму происходит; с уменьшением $d$ это приближение замедляется, но зато становится более эффективным. С ещё меньшими значениями параметра $d$ приближение к оптимуму также замедляется и становится ещё более эффективным (рис. 2). Например, при $d = 0.1$ $\rho_m$ уменьшается до величин порядка 0.02.

Итак, в нашей иллюстративной модели агент способен быть помощником людей и оптимизировать процесс выращивания пшеницы.

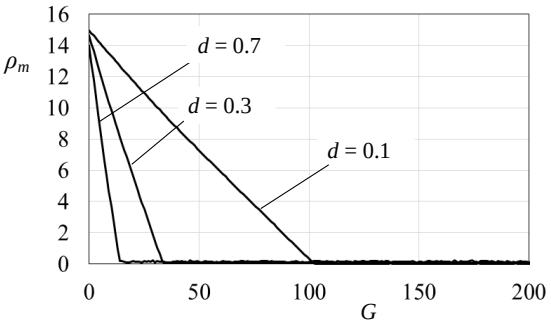



Рис. 2. Зависимости расстояния $\rho_m = \rho_{k\_min}(\mathbf{t}_{k\_min}(G), \mathbf{t}_{OPT})$ до оптимума для наилучшего агента от номера поколения $G$ для $d = 0.1, 0.3, 0.7$

# 7. Заключение

Конечно, способности рассмотренного агента ещё далеки от полноценного научного познания, но у него уже формируются начальные шаги к такому познанию. Такой агент имеет и использует определённые понятия: время, число, солнце, ветер, тучи. Агенту несложно формировать близкие понятия: пространство, предмет, движение. Он способен анализировать причинные связи между событиями в окружающем мире. Агент способен анализировать простые явления природы. Иллюстративный пример показывает, что такой агент способен выполнять полезную работу.

Отметим возможности дальнейшего развития процессов познания когнитивным агентом. На наш взгляд, наиболее важно проанализировать пути, ведущие к научному познанию природы. Как отмечено в книге [4], важная способность, которая возникла на заре возникновения научного познания – критическое мышление, которое отличается от первобытного мышления тем, что возникает оценка мыслительного процесса самим мыслящим субъектом [4, гл. 8]:

> «Критическое мышление рассматривает каждое объяснение (языковую модель действительности) наряду с другими, конкурирующими объяснениями (моделями), и оно не удовлетворится, пока не будет показано, чем данное объяснение лучше, чем конкурирующее».

В работе [7] проанализирована простая модель процесса формирования доказательства на основе использования критического мышления. Дальнейшее развитие научного познания должно включать возникновение аксиоматического метода. В работе [8] проанализирован вопрос: может ли компьютерный автономный агент сам «изобрести» аксиоматический метод и применить его в определенной математической теории?

Конечно, работы [7, 8] и настоящая модель

характеризуют только отдельные элементы пути к пониманию процессов возникновения и формирования научного познания. Эти работы, как и книги [1, 4] развивают основы моделирования когнитивной эволюции.

Настоящая работа выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2024-0001 «Создание и реализация доверенных систем искусственного интеллекта, основанных на новых математических и алгоритмических методах, моделях быстрых вычислений, реализуемых на отечественных вычислительных системах» (1023032100070-3-1.2.1). Автор благодарен З.Б. Соховой за плодотворные дискуссии.

# Cognition and Use of the Properties of Nature by a Cognitive Autonomous Agent

**Vladimir G. Red'ko**

**Abstract.** A model of a cognitive autonomous agent that cognizes simple properties of nature is constructed and analyzed. It is assumed that the agent develops the ability of primitive thinking; in particular, the agent has the ability to predict future events in the external world and has an elementary feeling of causality. The agent forms concepts characterizing the external world, observes and analyzes phenomena of the external world. An illustrative computer model of a cognitive autonomous agent using its abilities to perform certain useful work has been constructed and analyzed.

**Keywords:** cognitive autonomous agent, simple forms of thinking, cognition of elementary properties of nature, scientific cognition

## Литература

1. В.Г. Редько. Моделирование когнитивной эволюции: На пути к теории эволюционного происхождения мышления. Изд. 2, испр. и доп. М., URSS, 2019.

2. Л. Леви-Брюль. Первобытное мышление. М., Атеист, 1930.

3. Л. Леви-Брюль. Первобытный менталитет. СПб., «Европейский Дом», 2002.

4. В.Ф. Турчин. Феномен науки: Кибернетический подход к эволюции. М., Наука, 1993. См. также http://www.refal.ru/turchin/phenomenon/ (дата обращения 06.03.2024).

5. Д. Юм. Исследование о человеческом познании. Соч. в 2-х томах, т. 2. М., Мысль, 1966, 5–169.

6. В.Г. Редько. Модель чувства причинности. «Труды НИИСИ РАН», Т.10 (2020), № 2, 34–38.

7. В.Г. Редько. От критических методов к процессам доказательств. «Труды НИИСИ РАН», Т.13 (2023), № 4, 123–126.

8. В.Г. Редько. Как автономный когнитивный агент может создавать аксиоматическую теорию. «Труды НИИСИ РАН», Т.13 (2023), № 1-2, 46–51.

# Модели взаимодействующих автономных агентов

**В.Г. Редько**

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, vgredko@gmail.com

**Аннотация.** В данной работе исследованы модели взаимодействующих автономных агентов. В основном изучаются биологически инспирированные модели, характеризующие взаимодействие, в результате которого агенты обеспечивают охрану собственной территории. В частности, в компьютерной модели было показано, что два агента, помещённые в клеточный мир, способны разделить этот мир на два примерно равных участка, каждый участок представляет собой «собственность» одного агента. Отдельный агент охраняет свой участок, угрожая другому агенту. Также проанализированы случаи коллективной защиты относительно слабыми агентами своей территории от хищников.

**Ключевые слова:** взаимодействующие автономные агенты, охрана своей территории, агрессивность, коллективное поведение

## 1. Введение

В настоящей работе изучаются модели взаимодействующих автономных агентов. Рассматриваются биологически инспирированные модели, характеризующие взаимодействие, в результате которого агенты обеспечивают охрану собственной территории. При исследовании моделей будем отталкиваться от работ одного из основателей этологии (науки о поведении животных) Конрада Лоренца, который детально характеризует взаимодействие животных при охране своей территории [1, 2]. При охране своей территории агенты часто проявляют определённую агрессивность.

## 2. Биологически инспирированные модели охраны собственной территории

### 2.1. Биологические примеры охраны животными своей территории, примеры агрессии

Приведём характерные примеры охраны животными своей территории, примеры агрессивности, представленные в [1, 2].

Большие стаи скворцов коллективно нападают в воздухе на ястреба-перепелятника, отбивая у него возможность нападать на них. Отметим, что примерно такое поведение (нападение большого коллектива слабых агентов на агентов-«ястребов») возникало при компьютерном моделировании эволюции агентов, имеющих признаки близких по виду агентов [3-5].

Аналогично гуси могут сомкнутыми рядами атаковать лису, которая медленно отступает от трубящей стаи гусей.

Ещё один яркий пример. Лоренц описывает, как он плавал с маской, дыхательной трубкой и ластами в море в окрестностях Флориды [1, 2]. Он наблюдал, что в поле его зрения было только по одному экземпляру некоторых ярко окрашенных рыб. А когда одна из таких рыб стала медленно приближаться к наблюдательному пункту Лоренца (по-видимому, в поисках пищи), то местная рыба того же вида с беспримерной яростью бросилась на чужака и прогнала его со «своей территории». Причём рыб другого вида «хозяин» участка не трогал. Лоренц интерпретировал это как борьбу «хозяина» за «свой» участок добычи пищи.

Примерно такое же поведение наблюдалось в большом аквариуме для двух рыб (цихлид). Сначала более сильная рыба постаралась захватить весь аквариум (как свою собственную территорию), а вторую загнала в маленький уголок. Но потом вторая рыба постепенно расширяла свой участок и в результате две рыбы смогли поделить аквариум на два примерно равных участка, после чего два угрожающие друг другу противника непрерывно патрулировали вдоль границы между этими участками. Отметим, что здесь мы немного упрощённо (но достаточно для построения модели) характеризуем основные черты эксперимента с рыбами в большом аквариуме, подробнее см. [1, 2].

Если аквариум не очень просторный, то происходит агрессивное нападение рыб друг на друга. При этом самцы более агрессивные, чем самки. И даже если в аквариуме остаётся одна пара рыб (самец и самка), то самец может растерзать самку.

Интересный эксперимент проделали с двумя парами рыб, разделив один достаточно большой аквариум пополам прозрачным стеклом и поместив каждую пару в свою половину аквариума. Тогда каждая рыба направляла свою здоровую

злость на соседа своего пола: самка нападала на самку, а самец – на самца. Но иногда в таком эксперименте пограничное стекло зарастало водорослями, и тогда в каждой половине аквариума самец начинал грубо обращаться со своей самкой. Стоило как следует протереть разделительное стекло между «квартирами», как восстанавливалась яростная, но безвредная ссора между соседями через прозрачное стекло.

Подробнее о таких примерах агрессивности см. [1, 2].

## 2.2. Описание модели разделения территории

Отталкиваясь от примера разделения аквариума двумя рыбами, построим модель двух автономных агентов, разделяющих общую территорию. Причём будем рассматривать: а) случай большой территории, на которой, в конце концов, могут разместиться два агента, б) случай относительно малой территории, которой недостаточно для двух размещения двух агентов.

*Свойства территории и агентов.* Будем считать, что территория и агенты обладают следующими свойствами. Территория (мир агентов) ограничена, считаем, мир агентов – полоска местности определённой длины, состоящая из отдельных клеток. То есть мир агентов – это цепочка клеток. На этой территории имеется пища агентов, порции пищи случайно распределены по клеткам территории. В одной клетке может быть только одна порция пищи. Число клеток в мире агентов равно $N$. Число порций пищи равно $M$ ($M < N$). Агенты могут съедать эти порции пищи. Когда агент съедает порцию пищи в определённой клетке, то новая порция пищи появляется в другой случайной клетке, в которой ещё нет порции пищи. То есть количество порций пищи в мире постоянно.

Оба агента имеют определённый ресурс $R_1$ и $R_2$. Кроме того, у каждого агента имеется «свой» участок, аналог собственного дома. При качественном рассмотрении поведения агентов можно считать, что сначала первый агент имеет больший ресурс: $R_1 > R_2$. При компьютерных расчетах обычно считалось, что начальные ресурсы агентов одинаковы. Ресурсы агентов меняются со временем. Считаем, что время дискретно, $t = 1, 2\ldots$. В начальный момент времени первый агент считает своим участком весь мир, кроме самой крайней правой клетки, на которую с помощью угроз он вытесняет второго агента (это аналогично изложенному выше эксперименту с двумя рыбами в большом аквариуме).

Опишем подробнее *действия агентов*. Каждый такт времени каждый агент может выполнить следующие действия:

1) Угрожать другому агенту.
2) Нанести удар другому агенту, это можно сделать, когда этот другой агент находится в соседней клетке.
3) Съесть порцию пищи, если в той клетке, в которой находится агент, имеется порция пищи.
4) Уйти от угрожающего агента в противоположную от него сторону.
5) Переместиться на одну свободную клетку случайно вправо или влево, если текущая клетка агента не находится на краю мира, а если текущая клетка находится на краю мира, то переместиться на одну клетку в сторону от края мира.
6) Находиться в состоянии покоя, не перемещаться.

Считаем, что *качественно поведение агентов в большом мире* можно представить следующим образом.

После того как первый агент, угрожая второму, вытеснил второго агента на крайнюю правую клетку, первый агент начинает ходить по своему участку и питается, находя порции пищи. Второй агент делает попытки выйти из своей правой клетки, но видя такие попытки, первый агент угрожает второму. После такой угрозы второй агент отступает на свою правую клетку. Тем не менее, спустя некоторое время второй агент снова повторяет попытки пройти левее. Постепенно второй агент отвоёвывает у первого сначала одну клетку, а затем после повторных попыток может постепенно отвоевать и другие клетки. Считаем, что клетка становится принадлежащей к участку второго агента после того, как второй агент побывал на этой клетке не менее $K$ раз. При этом первый агент после каждой попытки второго агента выйти из своего участка в левую сторону угрожает второму агенту. Когда второй агент вносит определённую клетку в свой участок, то участок первого агента уменьшается на эту клетку. Такой итеративный процесс перенесения пограничных клеток из участка первого агента в участок второго агента продолжается до тех пор, пока размеры участков агентов не станут приближённо равны друг другу. Когда какой-либо агент приближается к участку другого агента, то этот другой агент угрожает тому, который приближается.

Для количественной характеристики участков в итеративном процессе можно ввести границу участка второго агента, т.е. крайнюю левую клетку участка второго агента. Обозначим номер этой клетки $C_B$.

При съедании агентом порции пищи, его ресурс увеличивается на величину $D_P$. При выполнении любого из других действий ресурс агента уменьшается на величину $D_{M1}$, $D_{M2}$, $D_{M4}$, $D_{M5}$, $D_{M6}$, соответственно. При получении удара ре-

сурс ударяемого агента уменьшается на величину $D_L$. Считаем, что потеря ресурса при получении удара значительно больше, что при остальных действиях.

Если ресурс агента стал меньше нуля, то такой агент погибает.

*Выбор действия* осуществляется следующим образом. Перечисляем действия в порядке их приоритетов.

А. Если к рассматриваемому агенту приблизился другой агент на расстояние, меньшее, чем *Dist*, то рассматриваемый агент угрожает другому. Это действие можно обозначить «угрожать другому агенту». Если агент выполнил действие «угрожать», то он уже не выполняет других действий.

Б. Если агент видит другого угрожающего агента, то агент либо перемещается на одну клетку в противоположную сторону от угрожающего агента, если имеется свободная соседняя клетка, расположенная в противоположную от угрожающего агента сторону, либо остаётся в состоянии покоя, если такой свободной клетки нет. Это действие можно обозначить «уйти от угрожающего агента».

В. Если агент не выбрал действия согласно пунктам А, Б, то он проверяет, имеется ли порция пищи в его клетке. Если в клетке имеется пища, то агент съедает всю порцию пищи из своей клетки. Это действие можно обозначить «питаться».

Г. Если агент не выбрал никакого действия согласно пунктам А-В, и нет другого агента в ближайших трёх клетках, то агент выбирает действие «случайно переместиться». При этом агент перемещается на одну свободную клетку случайно вправо или влево, если текущая клетка агента не находится на краю мира, а если текущая клетка находится на краю мира, то он перемещается на одну клетку в сторону, противоположную этому краю мира.

Д. Если агент не выбрал никакого действия согласно пунктам А-Г, то он выбирает действие «покой».

Каждый такт времени агент выполняет одно действие. В конце каждого такта проверяется, вышел ли второй агент из своего участка и, если он из участка вышел, то подсчитывается, сколько раз этот агент посетил эту новую для него клетку. Если число посещений этой клетки превысило определённый порог $K$, то второй агент присоединяет эту клетку к своему участку, а участок первого агента уменьшается на эту клетку, в результате граница между участками смещается влево, т.е. величина $C_B$ уменьшается на 1.

Для случая большой территории не будем рассматривать удары между агентами. Удары будем рассматривать в случае малой территории.

*В случае малого мира* (содержащего, например, всего 4 клетки) первый агент быстро загоняет второго агента в правый край мира, после чего агенты начинают ударять друг друга и терять ресурс. Так как ресурс второго агента меньше, чем ресурс первого, то ресурс второго агента уменьшается до нуля, и второй агент погибает. Остается только первый агент. Так как такое поведение качественно понятно, то компьютерное моделирование для случая малого мира не проводилось.

Для детального анализа поведения агентов в случае большого мира было проведено компьютерное моделирование.

## 2.3. Результаты компьютерного моделирования. Случай большой территории

Приведём результаты компьютерного моделирования.

Расчёты проводились для следующих параметров.

Число клеток в мире агентов $N = 20$. Число порций пищи $M = 10$. Порог для присоединения клетки к участку второго агента $K = 2$. Минимальное расстояние между агентами, при котором агент решает угрожать другому агенту, равно $Dist = 5$. Прирост ресурса агента ($R_1$ или $R_2$) при съедании им порции пищи равен $D_P = 1$. Расход ресурса агента на действие «угрожать другому агенту» равен $D_{M1} = 0.02$, расход ресурса на действия «уйти от угрожающего агента» и «случайно переместиться» равен $D_{M4} = D_{M5} = 0.01$. На действие «покой» ресурс не расходовался. Начальный ресурс агентов был равен 10.

На рис. 1-3 показаны примеры динамики агентов для различных используемых датчиков случайных чисел.

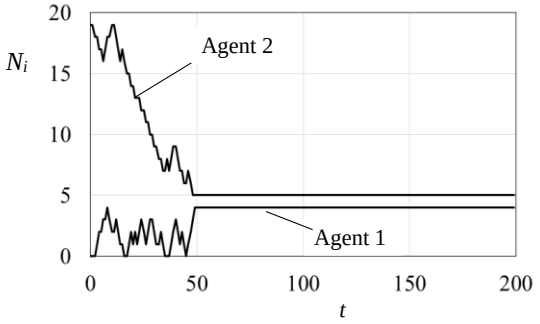



Рис. 1. Пример зависимости положения агентов (номеров клеток $N_i$) от времени $t$. Отдельный расчёт

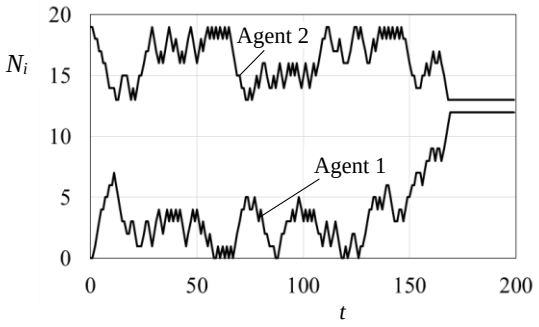



Рис. 2. Пример зависимости положения агентов (номеров клеток $N_i$) от времени $t$. Отдельный расчёт

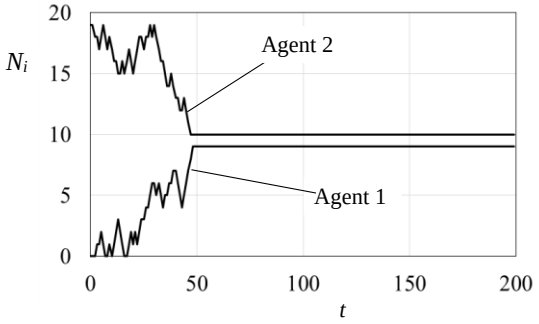



Рис. 3. Пример зависимости положения агентов (номеров клеток $N_i$) от времени $t$. Отдельный расчёт

Приведённые рисунки показывают, что, хотя конкретная динамика для этих случаев несколько различна, тем не менее, для всех случаев характерны следующие общие черты. Первый агент начинает двигаться из нулевой клетки, второй агент – из 19-й клетки; в конечном итоге, агенты останавливаются на удалении одной клетки друг от друга. Анализ действий агентов показывает, что в конечных положениях агенты только угрожают друг другу.

Для более чёткого анализа было проведено усреднение по 10000 различных вариантов датчиков случайных чисел. Усреднённые результаты представлены на рис. 4, 5.

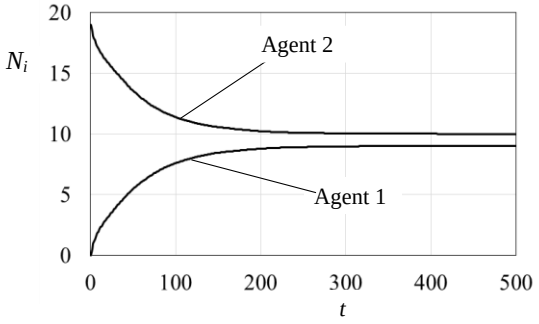



Рис. 4. Усреднённые зависимости положения агентов (номеров клеток $N_i$) от времени $t$

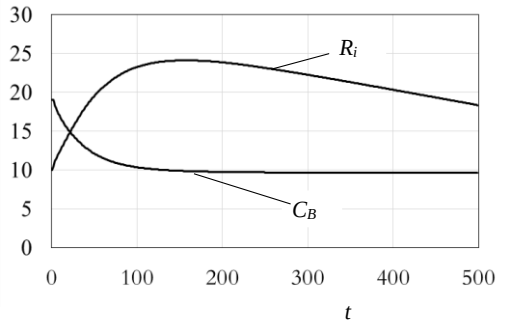



Рис. 5. Усреднённые зависимости ресурса агентов $R_i$ и номера пограничной клетки между участками агентов $C_B$ от времени $t$

Рис. 4 показывает, что агенты сначала движутся из крайних клеток мира навстречу друг другу. Наблюдение за действиями агентов показывает, что сначала они могут случайно перемещаться, питаться, угрожать друг другу и уходить от угрожающего агента (один от другого). А в конце концов, агенты останавливаются в среднем на расстоянии одной клетки друг от друга в середине мира, при этом они только угрожают друг другу и не совершают других действий.

Рис. 5 показывает, как меняются ресурсы агентов $R_1$ и $R_2$ (в среднем эти ресурсы для двух агентов приближённо равны друг другу, поэтому зависимость ресурсов от времени показана одной кривой). Причём в начале процессов агенты эпизодически питаются и их ресурсы за счёт этого растут. Но в дальнейшем агенты только угрожают друг другу, а это действие приводит к расходу ресурсов и ресурсы агентов постепенно уменьшаются.

Отметим, что однозначный выбор действия «угрожать» в конце процессов обусловлен схемой строгих приоритетов действий (см. пункты А-Д предыдущего раздела). Такую приоритетность можно немного ослабить, если модифицировать схему выбора действия агентом следующим образом. Считаем, что агент совершает определённое действие не каждый раз, когда выполняются перечисленные в пунктах А-Д условия, а с определённой вероятностью при выполнении условий этой схемы. Тогда в конце рассматриваемых процессов агенты будут выполнять не только действие «угрожать», а иногда и другие действия, в частности, эпизодически они будут питаться. Такое поведение агентов при ослаблении приоритетности было подтверждено компьютерными расчётами. Причём расчёты показали, что при достаточном ослаблении приоритетности ресурсы агентов в конце процессов не уменьшаются.

На рис. 5 приведена также усреднённая зависимость от времени номера пограничной клетки между участками агентов $C_B$ . Видно, что в конце

процессов граница между участками приближается к середине мира.

Таким образом, результаты компьютерного моделирования (рис. 4, 5) показывают, что агенты делят клеточный мир на два примерно равных участка, и в конце концов, останавливаются на границах участков и угрожают друг другу. Такое поведение согласуется с результатом биологического эксперимента с двумя рыбами в большом аквариуме [1, 2].

### 2.4. Коллективная защита своей территории

Перейдём к рассмотрению свойств коллективной защиты своей территории от хищников. Биологические примеры такого поведения (защита большой стаи скворцов от ястреба-перепелятника и защита стаи гусей от лисы [1, 2]) были представлены выше в разделе 2.1.

Такое поведение целого коллектива достаточно понятно. Тем не менее, возникает важный вопрос: как может зародиться такое поведение? Определённый ответ на этот вопрос дают работы [3-5], в которых рассматривалась эволюция популяции агентов в двумерной клеточной среде и эволюционно возникала коллективная защита агентов от хищников.

Кратко рассмотрим модель работ [3-5]. Агенты эволюционирующей популяции могли выполнять следующие действия: отдыхать, питаться, рождать потомка, переместиться на одну из соседних клеток, атаковать другого агента в своей клетке. Сенсоры агента воспринимали уровень собственной внутренней энергии, наличие ресурса и число агентов в ближайших клетках. Кроме того, каждый агент имел маркер, кодируемый вектором достаточно большой размерности. Маркеры служили индикаторами схожести агентов. Поведение агентов управлялось однослойной нейронной сетью. Веса синапсов нейронной сети и вектор-маркер составляли генотип агента. В процессе эволюции генотип передавался от родителей к потомкам с малыми мутациями. Агенты с близкими маркерами представляли собой группы «своих» агентов, т.е. своего рода «родственников». В частности, агент-потомок имел маркер, близкий к маркеру агента-родителя. Расстояние между собственным маркером агента и маркером другого агента в клетке служило отдельным входом нейронной сети агента. То есть, благодаря маркерам, агенты могли отличать агентов своей группы от агентов чужой группы (родственников от не-родственников).

Результаты компьютерного моделирования работ [3-5] показали, что в процессе проанализированной эволюции происходит формирование ряда стратегий поведения агентов. Было обнаружено эволюционное формирование известных стратегий поведения, аналогов голубей, ястребов и буржуа [6]. «Голуби» никогда не атакуют других агентов и пытаются уйти из-под атаки, а «ястребы» охотятся на других агентов. «Буржуа» находятся в одной и той же клетке и атакуют любого чужого агента, входящего в эту клетку.

Кроме того, формировались ещё две новые стратегии: «вороны» и «скворцы». Ворон покидал клетку с вместе членами своей группы (своими родственниками), но при обнаружении в клетке агента, не принадлежащего к своей группе, атаковал его. Скворцы оставались в клетке с членами своей группы и коллективно нападали на любого чужака, попадающего в клетку. Каждый скворец имел малый ресурс, однако коллективно скворцы могли уничтожить чужака, благодаря численному перевесу. Стратегия скворцов как раз подобна стайной защите от хищника, охарактеризованной в работах [1, 2]. Таким образом, в работах [3-5] в процессе эволюции агентов формировалась кооперативная защита своей территории агентами от «хищников». Важное свойство этих агентов – наличие маркеров, благодаря которым агенты могли отличать агентов своей группы от чужих агентов.

## 3. Заключение

Таким образом, в настоящей работе изучены биологически инспирированные модели защиты своей территории автономными агентами. В частности, построена и исследована компьютерная модель двух автономных агентов, разделяющих общую территорию. Результаты компьютерного моделирования показывают, что эти агенты делят клеточный мир на два примерно равных участка, и в конце концов, останавливаются на границах своих участков и угрожают друг другу. Это прямо согласуется с результатами биологического эксперимента с двумя рыбами в большом аквариуме [1, 2].

Проанализированы процессы возникновения коллективной защиты относительно слабыми агентами своей территории от хищников.

Настоящая работа выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2024-0001 «Создание и реализация доверенных систем искусственного интеллекта, основанных на новых математических и алгоритмических методах, моделях быстрых вычислений, реализуемых на отечественных вычислительных системах» (1023032100070-3-1.2.1). Автор благодарен Б.В. Крыжановскому за плодотворные дискуссии.

# Models of Interacting Autonomous Agents

**Vladimir G. Red'ko**

**Abstract.** In this work, models of interacting autonomous agents are studied. Biologically inspired models are studied; agents ensure the protection of their own territory in these models. In particular, it is shown that two agents placed in a cellular world are capable to divide this world into two approximately equal sections, each section representing the "property" of one agent. Each agent protects its section, threatening the other agent. Cases of collective defense by relatively weak agents of their territory from predators are also analyzed.

**Keywords:** interacting autonomous agents, protection of a territory, aggressiveness, collective behavior

## Литература

1. К. Лоренц. Агрессия, или Так называемое зло. М., АСТ, 2023.
2. К. Лоренц. Так называемое зло. М., Культурная революция, 2011.
3. M. Burtsev, P. Turchin. Evolution of cooperative strategies from first principles. "Nature" V. 440 (2006), № 7087, 1041–1044.
4. М.С. Бурцев, П.В. Турчин. Эволюция кооперативных стратегий из первых принципов. В книге П.В. Турчин. Историческая динамика. На пути к теоретической истории. М., Издательство ЛКИ, 2007. С. 317–328. См. также: https://www.keldysh.ru/pages/mrbur-web/publ/burtsev.turchin.(2007).nat_rus.pdf (дата обращения 12.03.2024).
5. М.С. Бурцев. Исследование новых типов самоорганизации и возникновения поведенческих стратегий. Кандидатская диссертация. Институт прикладной математики им. М.В. Келдыша РАН, Москва, 2005, https://www.keldysh.ru/pages/mrbur-web/diss/ (дата обращения 12.03.2024).
6. J. Maynard Smith. The theory of games and the evolution of animal conflicts. "Journal of Theoretical Biology" V. 47 (1974), 209–222.

Подписано в печать  29.03.2024 г.
Формат 60x90/8
Печать цифровая. Печатных листов 5.5
Тираж 100 экз. Заказ  №  233

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