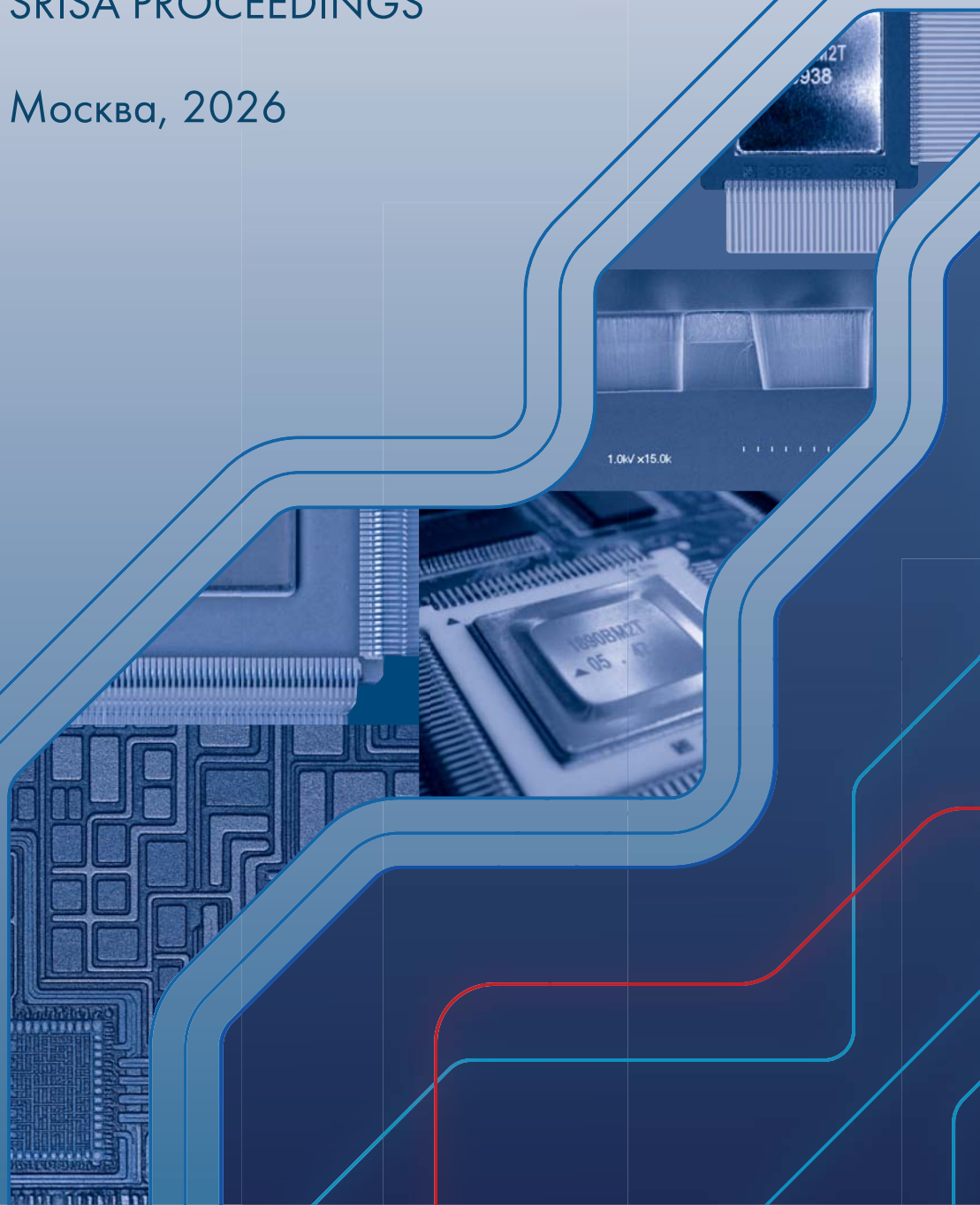


Труды НИИСИ

SRISA PROCEEDINGS

Москва, 2026



Федеральное государственное бюджетное учреждение «Федеральный научный центр
Научно-исследовательский институт системных исследований
Национального исследовательского центра «Курчатовский институт»
(НИИЦ «Курчатовский институт» — НИИСИ)

ТРУДЫ НИИСИ

SRISA PROCEEDINGS

ТОМ 16 № 2

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ
МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2026

Учредитель и издатель

Федеральное государственное бюджетное учреждение «Федеральный научный центр
Научно-исследовательский институт системных исследований
Национального исследовательского центра «Курчатовский институт»
(НИЦ «Курчатовский институт» — НИИСИ)

«Труды НИИСИ» — это рецензируемый научный журнал, в котором публикуются научные статьи по следующим специальностям и отраслям наук:

- 1.2.1. «Искусственный интеллект и машинное обучение» (физико-математические науки);
- 2.3.1. «Системный анализ, управление и обработка информации, статистика» (физико-математические и технические науки);
- 2.3.2. «Вычислительные системы и их элементы» (технические науки);
- 2.3.3. «Автоматизация и управление технологическими процессами и производствами» (технические науки);
- 2.3.5. «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей» (физико-математические и технические науки).

Направления исследований, по которым журнал публикует оригинальные статьи определены, но не ограничены следующим перечнем: системный анализ; математика; математическое и компьютерное моделирование; задачи автоматизации и управления; обработка сигналов; компьютерное зрение и обработка изображений; распознавание образов; статистика; технологии искусственного интеллекта; информационные технологии; информационная безопасность; вычислительные системы и их элементы; микро- и наноэлектроника; высокопроизводительные вычисления; вопросы численного анализа; нейроморфные и мягкие вычисления; оптико-нейронные технологии; история науки, техники и персоналий. Журнал предназначен для научных сотрудников, инженеров и аспирантов, работающих в указанных направлениях исследований.

Миссия журнала — развитие перечисленных научных направлений в России и за рубежом, включая широкое освещение результатов исследований и обеспечение высококвалифицированных кадров печатными площадями, обеспечение высокого качества исследований путем развития механизма профессионального и общественного обсуждения научных результатов и воспитания молодого поколения ученых-исследователей.

Политика журнала ориентирована на пропаганду передовых научно-технических идей и решений в рамках развития важнейших наукоемких технологий и участия в реализации приоритетов научно-технологического развития Российской Федерации. До 2025 года журнал издавался в печатном виде с названием «Труды НИИСИ РАН» (ISSN 2225-7349). В настоящее время сетевому рецензируемому научному журналу «Труды НИИСИ» присвоен ISSN 3033-6422.

Главный редактор

Бетелин Владимир Борисович, академик РАН, д. ф.-м. н., профессор,
научный руководитель НИЦ «Курчатовский институт» — НИИСИ, Москва

Заместители главного редактора

Крыжановский Борис Владимирович, чл.-корр. РАН, д.ф.-м.н., главный научный сотрудник центра оптико-нейронных технологий НИЦ «Курчатовский институт» — НИИСИ, Москва

Шабанов Борис Михайлович, чл.-корр. РАН, д.т.н., доцент, руководитель отделения суперкомпьютерных систем и параллельных вычислений НИЦ «Курчатовский институт», Москва

Члены редакционной коллегии

Аветисян Арутюн Ишханович, академик РАН, д.ф.-м.н., профессор, директор ИСП РАН, Москва

Панченко Владислав Яковлевич, академик РАН, д. ф.-м. н., профессор,
вице-президент РАН, Москва

Савин Геннадий Иванович, академик РАН, д. ф.-м. н., профессор,
научный руководитель МСЦ — филиала НИЦ «Курчатовский институт» — НИИСИ, Москва

Сигов Александр Сергеевич, академик РАН, д.ф.-м.н., профессор,
президент РТУ МИРЭА, Москва

Бланк Владимир Давыдович, д.ф.-м.н., профессор,
и.о. директора НИЦ «Курчатовский институт» — ТИСНУМ, Троицк

Галкин Валерий Алексеевич, д.ф.-м.н., профессор,
директор Сургутского филиала НИЦ «Курчатовский институт» — НИИСИ, Сургут

Куклин Владимир Жанович, д.т.н., доцент, ведущий научный сотрудник лаборатории автоматизации
и управления технологическими процессами НИЦ «Курчатовский институт» — НИИСИ, Москва

Леонов Александр Георгиевич, д.п.н., к.ф.-м.н., доцент,
ведущий научный сотрудник лаборатории вычислительных методов
механико-математического факультета МГУ им. М.В. Ломоносова, Москва

Михайлюк Михаил Васильевич, д.ф.-м.н., профессор, главный научный сотрудник отдела
программных средств визуализации НИЦ «Курчатовский институт» — НИИСИ, Москва

Олейник Андрей Владимирович, д.т.н., профессор,
заместитель директора по стратегическому развитию ИКТИ РАН, Москва

Пархоменко Юрий Николаевич, д.ф.-м.н., профессор, научный руководитель и профессор кафедры
материаловедения полупроводников и диэлектриков НИТУ МИСиС, Москва

Редько Владимир Георгиевич, д.ф.-м.н., с.н.с., главный научный сотрудник центра оптико-
нейронных технологий НИЦ «Курчатовский институт» — НИИСИ, Москва

Смирнов Николай Николаевич, д.ф.-м.н., профессор, заведующий лабораторией волновых
процессов механико-математического факультета МГУ им. М.В. Ломоносова, Москва

Сотников Александр Николаевич, д.ф.-м.н., профессор, г.н.с. отделения суперкомпьютерных
систем и параллельных вычислений НИЦ «Курчатовский институт» — НИИСИ, Москва

Шелепин Николай Алексеевич, д.т.н., профессор,
руководитель научного направления «Микроэлектроника» ИНМЭ РАН, Москва

Александров Ислам Александрович, к.т.н., доцент, заместитель директора по научной и
методической работе НИЦ «Курчатовский институт» — НИИСИ, Москва

Аряшев Сергей Иванович, к.т.н., заместитель директора по микроэлектронике и
вычислительным системам НИЦ «Курчатовский институт» — НИИСИ, Москва

Годунов Александр Николаевич, к.ф.-м.н., с.н.с., заведующий отделом системного
программирования НИЦ «Курчатовский институт» — НИИСИ, Москва

Грюнталь Андрей Игоревич, к.ф.-м.н., заведующий отделом математического обеспечения
НИЦ «Курчатовский институт» — НИИСИ, Москва

Карандашев Яков Михайлович, к.ф.-м.н., ведущий научный сотрудник центра оптико-нейронных
технологий НИЦ «Курчатовский институт» — НИИСИ, Москва

Кушиниренко Анатолий Георгиевич, к.ф.-м.н., доцент,
заведующий отделом учебной информатики НИЦ «Курчатовский институт» — НИИСИ, Москва

Мамросенко Кирилл Анатольевич, к.т.н., доцент, в.н.с., руководитель центра визуализации и
спутниковых информационных технологий НИЦ «Курчатовский институт» — НИИСИ, Москва

Муранов Александр Николаевич, к.т.н., доцент, заведующий лабораторией автоматизации и
управления технологическими процессами НИЦ «Курчатовский институт» — НИИСИ, Москва

Петров Константин Александрович, к.т.н., старший научный сотрудник отдела архитектур
высокопроизводительных микропроцессоров НИЦ «Курчатовский институт» — НИИСИ, Москва

Семенов Илья Витальевич, к.ф.-м.н., ведущий научный сотрудник отдела вычислительной
математики НИЦ «Курчатовский институт» — НИИСИ, Москва

Цимбалов Андрей Сергеевич, к.т.н., заместитель директора по микротехнологии
НИЦ «Курчатовский институт» — НИИСИ, Москва

Founder and Publisher

Scientific Research Institute for System Analysis of the National Research Center "Kurchatov Institute" (NRC "Kurchatov Institute" - SRISA)

SRISA Proceedings is a peer-reviewed journal that covers the following key areas of research:

- Artificial intelligence and machine learning
- System analysis, control, information processing, statistics
- Computing systems and their components
- Automation and control in manufacturing
- Mathematical and software support for computing systems, complexes and computer networks.

We publish original articles on topics including, but not limited to: system analysis; mathematics; computer simulation; automation and control; signal processing; computer vision and image processing;

pattern recognition; statistics; artificial intelligence; information technologies; cybersecurity; computing systems and their components; micro- and nanoelectronics; high-performance computing; numerical analysis; neuromorphic and soft computing; optoneural technologies; and the history of science, technology, and researchers. Our readers include researchers, engineers, and doctoral students.

Our mission is to advance these research areas in Russia and worldwide by publishing significant results and offering leading professionals a platform to share their work. We are committed to maintaining high research standards through professional and public review while fostering the next generation of researchers.

The journal's policy is to promote advanced research and innovative solutions, foster the development of high-tech fields, and contribute to key national priorities in science and technology. Until 2025, the journal was published in print as Trudy NIISI RAN (Proceedings of SRISA, Russian Academy of Sciences), ISSN 2225-7349. Currently, SRISA Proceedings online peer-reviewed journal has been assigned ISSN 3033-6422.

Chief Editor

Vladimir B. Betelin, member of the Russian Academy of Sciences, Doctor of Science (Phys&Math),
Prof., Academic Director, NRC "Kurchatov Institute" - SRISA, Moscow

Deputy Chief Editor

Boris V. Kryzhanovsky, corresponding member of the Russian Academy of Sciences,
Doctor of Science (Phys&Math), Chief Researcher, Center for Optic-Neural Technologies,
NRC "Kurchatov Institute" - SRISA, Moscow

Boris M. Shabanov, corresponding member of the Russian Academy of Sciences, Doctor of Science
(Engineering), Assoc. Prof., Head of the Department of Supercomputer Systems and Parallel Computing,
NRC "Kurchatov Institute" - SRISA, Moscow.

Editorial Board

Arutyun I. Avetisyan, member of the Russian Academy of Sciences, Doctor of Science (Phys&Math),
Prof., Director of the Institute for System Programming, Russian Academy of Sciences, Moscow
Vladislav Ya. Panchenko, member of the Russian Academy of Sciences, Doctor of Science (Phys&Math),
Prof., Vice-President of the Russian Academy of Sciences, Moscow

Gennady I. Savin, member of the Russian Academy of Sciences, Doctor of Science (Phys&Math),
Academic Director, JSC, a Branch of the NRC "Kurchatov Institute" - SRISA, Moscow
Alexander S. Sigov, member of the Russian Academy of Sciences, Doctor of Science (Phys&Math), Prof.,
President of the MIREA - Russian Technological University, Moscow

Vladimir D. Blank, Doctor of Science (Phys&Math), Prof., Acting Director, National Research Center
Kurchatov Institute – Technological Institute for Superhard and Novel Carbon Materials, Troitsk

Valery A. Galkin, Doctor of Science (Phys&Math), Prof., Director,
Surgut Branch of the NRC "Kurchatov Institute" - SRISA, Surgut

Vladimir Zh. Kuklin, Doctor of Science (Engineering), Assoc. Prof., Leading Researcher,
Manufacturing Automation and Control Laboratory, NRC "Kurchatov Institute" - SRISA, Moscow
Alexander G. Leonov, Doctor of Science (Education), PhD (Phys&Math), Assoc. Prof., Leading Researcher
of the Computational Methods Laboratory, School of Mechanics and Mathematics,
Lomonosov Moscow State University, Moscow

Mikhail V. Mikhailyuk, Doctor of Science (Phys&Math), Prof., Chief Researcher,
Visualization Software Department, NRC "Kurchatov Institute" - SRISA, Moscow

Andrey V. Oleinik, Doctor of Science (Engineering), Prof., Deputy Director for Strategic Development,
Institute for Design-Technological Informatics of the Russian Academy of Sciences, Moscow

Yuri N. Parkhomenko, Doctor of Science (Phys&Math), Prof., Scientific Supervisor and Prof.,
Department of Materials Science for Semiconductors and Dielectrics, MISiS, Moscow

Vladimir G. Redko, Doctor of Science (Phys&Math), Senior Researcher, Chief Researcher,
Center for Optic-Neural Technologies, NRC "Kurchatov Institute" - SRISA, Moscow

Nikolay N. Smirnov, Doctor of Science (Phys&Math), Prof., Head of the Wave Processes Laboratory,
School of Mechanics and Mathematics, Lomonosov Moscow State University, Moscow

Alexander N. Sotnikov, Doctor of Science (Phys&Math), Prof., Head of the Department of Supercomputer
Systems and Parallel Computing, NRC "Kurchatov Institute" - SRISA, Moscow

Nikolay A. Shelepin, Doctor of Science (Engineering), Prof., Microelectronics Research Advisor,
Institute of Nanotechnology of Microelectronics, Russian Academy of Sciences, Moscow

Islam A. Alexandrov, PhD (Engineering), Assoc. Prof., Deputy Director for Research and Education,
NRC "Kurchatov Institute" - SRISA, Moscow

Sergei I. Aryashev, PhD (Engineering), Deputy Director for Microelectronics and Computer Systems,
NRC "Kurchatov Institute" - SRISA, Moscow

Alexander N. Godunov, PhD (Phys&Math), Senior Researcher, Head of the Department of
System Programming, NRC "Kurchatov Institute" - SRISA, Moscow

Andrei I. Griuntal, PhD (Phys&Math), Head of the Mathematics Department,
NRC "Kurchatov Institute" - SRISA, Moscow

Yakov M. Karandashev, PhD (Phys&Math), Leading Researcher, Center for Optic-Neural Technologies,
NRC "Kurchatov Institute" - SRISA, Moscow

Anatoly G. Kushnirenko, PhD (Phys&Math), Assoc. Prof., Head of IT for Education Department,
NRC "Kurchatov Institute" - SRISA, Moscow

Kirill A. Mamrosenko, PhD (Engineering), Assoc. Prof., Leading Researcher, Head of the Center for Vis-
ualization and Satellite Information Technologies, NRC "Kurchatov Institute" - SRISA, Moscow

Alexander N. Muranov, PhD (Engineering), Assoc. Prof., Head of the Manufacturing Automation and
Control Laboratory, NRC "Kurchatov Institute" - SRISA, Moscow

Konstantin A. Petrov, PhD (Engineering), Senior Researcher, High Performance
Microprocessor Architectures Department, NRC "Kurchatov Institute" - SRISA, Moscow

Ilya V. Semenov, PhD (Phys&Math), Leading Researcher,
Department of Computational Mathematics, NRC "Kurchatov Institute" - SRISA, Moscow

Andrey I. Tsimbalov, PhD (Engineering), Deputy Director for Microtechnology,
NRC "Kurchatov Institute" - SRISA, Moscow

Научный редактор номера / Scientific editor of the issue:
Кушниренко А.Г. / Kushnirenko A.G.

СОДЕРЖАНИЕ

I. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И МАШИННОЕ ОБУЧЕНИЕ	
<i>А. С. Караваева, А. М. Куликов, Н. С. Мартынов, К. А. Мащенко, К. К. Пчелин, Е. Д. Тарасюк.</i> Эволюция архитектуры нейросетевого модуля: поддержка мультимодальных сервисов в цифровой образовательной платформе Мирера	8
<i>A. S. Karavaeva, A. M. Kulikov, N. S. Martynov, K. A. Mashchenko, K. K. Pchelin, E. D. Tarasuk.</i> Evolution of the neural network module architecture: supporting multimodal services in the Mirera digital educational platform	18
II. СИСТЕМНЫЙ АНАЛИЗ, УПРАВЛЕНИЕ И ОБРАБОТКА ИНФОРМАЦИИ, СТАТИСТИКА	
<i>Ш. М. Магомедов.</i> Сравнительный анализ архитектур многоагентных дебатов крупных языковых моделей в задаче поддержки принятия решений.	26
III. ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ И ИХ ЭЛЕМЕНТЫ	
<i>И. А. Лафазан.</i> Обеспечение устойчивого электропитания микропроцессорной системы при подключении через длинный кабель.	32
<i>Е. А. Соловьева, А. В. Корниленко.</i> Блок приема данных от внешнего АЦП с поддержкой ПДП по интерфейсу SPI	43
<i>Е. С. Зайцева, А. Д. Лысак.</i> Параметризация блока подключения приемопередатчиков контроллера PCI Express для оптимизации использования ресурсов ПЛИС и автоматизации верификации	48
IV. АВТОМАТИЗАЦИЯ И УПРАВЛЕНИЕ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ И ПРОИЗВОДСТВАМИ	
<i>А. А. Полин, В. А. Горелов, В. Ж. Куклин, А. Ю. Сафронов.</i> Параметрический синтез допустимых профилей винтовых роторов SRM на основе генетического алгоритма	54
V. МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ, КОМПЛЕКСОВ И КОМПЬЮТЕРНЫХ СЕТЕЙ	
<i>А. Б. Бетелин.</i> Применение генератора отчетов для преобразования таблиц из формата текстового файла в формат электронной таблицы	63

CONTENT

I. ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING	
<i>A. S. Karavaeva, A. M. Kulikov, N. S. Martynov, K. A. Mashchenko, K. K. Pchelin, E. D. Tarasuk. Evolution of the neural network module architecture: supporting multimodal services in the Mirera digital educational platform [Original article in Rus.]</i>	8
<i>A. S. Karavaeva, A. M. Kulikov, N. S. Martynov, K. A. Mashchenko, K. K. Pchelin, E. D. Tarasuk. Evolution of the neural network module architecture: supporting multimodal services in the Mirera digital educational platform</i>	18
II. SYSTEM ANALYSIS, CONTROL, INFORMATION PROCESSING, STATISTICS	
<i>Sh. M. Magomedov. Comparative analysis of multi-agent debate architectures of large language models in decision support</i>	26
III. COMPUTING SYSTEMS AND HARDWARE COMPONENTS	
<i>I. A. Lafazan. Providing stable power supply to the printed circuit board of a microprocessor system when connected via a long cable</i>	32
<i>E. A. Soloveva, A. V. Kornilenko. Data Acquisition Module for an External SPI ADC with DMA Support</i>	43
<i>E. S. Zaytseva, A. D. Lyspak. Parametrization of the transceiver connection block of a PCI Express controller for FPGA resource optimization and verification automation</i>	48
IV. AUTOMATION AND CONTROL IN MANUFACTURING	
<i>A. A. Polin, V. A. Gorelov, V. Zh. Kuklin, A. Yu. Safronov. Parametric synthesis of admissible SRM screw-rotor profiles based on a genetic algorithm</i>	54
V. MATHEMATICAL AND SOFTWARE SUPPORT FOR COMPUTING SYSTEMS, COMPLEXES AND COMPUTER NETWORKS	
<i>A. B. Betelin. Using a report generator to convert tables from a text file format to a spreadsheet format</i>	63

УДК 004.8:37.018.46

Эволюция архитектуры нейросетевого модуля: поддержка мультимодальных сервисов в цифровой образовательной платформе Мирера

А. С. Караваева¹, А. М. Куликов², Н. С. Мартынов³, К. А. Машченко⁴,
К. К. Пчелин⁵, Е. Д. Тарасюк⁶

¹НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, aleksandrakaravaeva@yandex.ru;

²НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, kulikrch@yandex.ru;

³НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, nikolai.martynov@math.msu.ru;

⁴НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, МГУ им. М. В. Ломоносова, Москва, Россия, Государственный университет управления, Москва, Россия, kirill.mashchenko@niisi.ru;

⁵НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, pcelinkosta0@gmail.com;

⁶НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, ekaterina.tarasuk@math.msu.ru;

Аннотация. В работе рассматривается развитие архитектуры нейросетевого модуля цифровой образовательной платформы Мирера в связи с расширением круга применяемых моделей искусственного интеллекта. Показано, что исходное LLM-ядро оркестратора не позволяло без существенных переработок подключать модели иных модальностей. Описана универсальная мультимодальная архитектура, включающая абстракцию сервис-менеджера на основе единого контракта, конфигурируемые цепочки обработки запросов и унифицированный поток выходных артефактов. Приведены примеры интеграций. Показано, что предлагаемая архитектура обеспечивает подключение новых типов моделей без модификации ядра маршрутизатора и формирует основу для дальнейшего развития мультимодальных образовательных сценариев.

Ключевые слова: цифровая образовательная платформа, ЦОП Мирера, нейросетевой оркестратор, мультимодальные модели, микросервисная архитектура, LLM, OCR, RabbitMQ, AI-агенты, образовательные технологии

1. Введение

В последние годы большие языковые модели (LLM) стали одним из ключевых направлений практического внедрения искусственного интеллекта в образовательную сферу. Появление высокопроизводительных открытых архитектур, их доступность в форме готовых API-сервисов и снижение требований к аппаратной инфраструктуре благодаря методам квантизации сделали возможным переход от единичных экспериментов к систематическому встраиванию языковых моделей в цифровые образовательные платформы. Сегодня LLM применяются для автоматизированной генерации учебных заданий и тестовых вопросов, рецензирования решений обучающихся, построения интерактивных помощников по конкретным дисциплинам, персонализации образовательной траектории и снижения административной нагрузки

преподавателей [1]. Параллельно расширяется и круг сопутствующих моделей искусственного интеллекта: к языковым моделям подключаются системы оптического распознавания, обработки изображений, распознавания и синтеза речи, формируя инструменты для работы с мультимодальными сценариями в цифровых образовательных платформах.

В цифровой образовательной платформе Мирера [2] накоплен опыт интеграции моделей искусственного интеллекта в широкий круг учебных задач. В работе [3] обобщены критерии выбора больших языковых моделей под образовательные сценарии и приведена сравнительная характеристика применяемых в платформе архитектур, а также описаны конкретные сценарии их использования: от генерации заданий и тестовых вариантов до построения помощников студента и преподавателя. В работе [4] изложен опыт интеграции нейросетевого ассистента

обучающихся по программированию, способного анализировать решения студентов в реальном времени и формировать обучающие рекомендации непосредственно в среде платформы. В работе [5] представлена система автоматизированного формирования образовательных видеоматериалов на основе модульного цифрового двойника преподавателя, объединяющая модули автоматического распознавания речи (ASR), синтеза речи (TTS) и синхронизации движений губ (LipSync), указанная система демонстрирует существенное сокращение трудозатрат преподавателя на подготовку учебных видеоматериалов при сохранении их высокого качества. Помимо собственно нейросетевых сервисов, платформа развивает и смежные направления автоматизации учебного процесса, в частности, гибридный учёт посещаемости с QR-верификацией и видеоидентификацией [6], что в совокупности формирует целостную инфраструктуру интеллектуальной поддержки участников образовательного процесса.

Образовательный процесс по своей природе мультимодален: помимо текстовых формулировок задач и решений, в нём задействованы изображения и схемы из учебных материалов, аудиозаписи и видеозаписи лекций, рукописные конспекты, презентации преподавателей. По мере усложнения образовательных сценариев в цифровой платформе круг применяемых моделей искусственного интеллекта закономерно выходит за рамки чисто языковых: становятся востребованными системы оптического распознавания учебных материалов и рукописного текста, модели распознавания и синтеза речи и другие нейросетевые модули. Каждое из этих направлений требует моделей с собственной модальностью входных данных, специфической аппаратной конфигурацией (например, специализированных сетей для оптического распознавания и визуально-языковых моделей для документов сложной структуры) и собственной семантикой выходного результата – структурированного текста, аудиосигнала, графического представления. Реализация подобных сценариев в рамках единой образовательной платформы требует, чтобы ядро нейросетевого модуля было способно подключать модели разной природы без существенных переработок маршрутизации, контрактов взаимодействия с платформой и инфраструктуры обработки запросов.

Реализация описанных в работах [3; 4] сценариев потребовала создания самостоятельного нейросетевого модуля, взаимодействующего с ядром платформы

Мирера через очереди сообщений и выполняющего функции маршрутизации запросов, управления пулом моделей и сборки ответов по контракту с платформой. Изначально этот модуль проектировался под единственный класс задач – обработку текстовых запросов с применением LLM, что определило его архитектурную особенность: все цепочки обработки запросов жёстко предполагали, что в конце сценария работы стоит языковая модель, обработка вложений (документов, изображений) выполнялась локально внутри агента с применением библиотеки распознавания, привнося в монолит существенные аппаратные зависимости.

С появлением задач иной модальности обнаружилась несовместимость подобной архитектуры с расширенным кругом моделей. Подключение каждого нового типа сервиса в её рамках потребовало бы либо дублирования инфраструктуры маршрутизации, либо точечных модификаций ядра под специфику конкретного сервиса, оба пути ведут к нарастанию сложности и ухудшению поддерживаемости. Кроме того, размещение зависимостей распознавания внутри монолита не позволяло независимо масштабировать ресурсы под задачи распознавания и под LLM-инференс, имеющие существенно различающиеся профили нагрузки и аппаратные требования. (LLM-инференсом мы называем этап, на котором уже обученная LLM-модель обрабатывает пользовательский запрос и генерирует ответ без изменения своих внутренних параметров).

Данная статья описывает архитектурную эволюцию нейросетевого модуля платформы Мирера, направленную на универсальную поддержку мультимодальных сервисов. В разделе 2 рассматривается исходная LLM-ориентированная архитектура и её ограничения. В разделе 3 описывается переход к универсальной мультимодальной архитектуре с единым контрактом сервис-менеджера, конфигурируемыми цепочками и унифицированным потоком выходных артефактов. В разделе 4 приведены практические реализации возможностей новой архитектуры.

2. Исходная LLM-ориентированная архитектура

2.1 Двухфазная обработка запросов

Нейросетевой модуль платформы Мирера построен на принципе асинхронной обработки запросов через брокер сообщений RabbitMQ, использующий протокол AMQP [7]. Запросы от

платформы поступают в виде структурированных сообщений-конвертов в очередь приёма, ответы помещаются в очередь уведомлений и доставляются обратно платформе для отображения результата участникам учебного процесса. Между этими крайними точками обработка запроса разделена на две фазы, каждая из которых реализуется собственным циклом-потребителем сообщений.

В фазе приёма (Intake loop) маршрутизатор выполняет валидацию структуры входящего сообщения-конверта, загружает описание цепочки обработки соответствующей команды из конфигурационного файла, проверяет полезную нагрузку на соответствие контракту команды и помещает сообщение в очередь первого сервиса в цепочке. Эта фаза не выполняет вычислительной работы и обеспечивает строгий входной контроль: некорректные запросы с нарушенной структурой сообщения, неизвестным именем команды или несоответствующей контракту полезной нагрузкой отсекаются до того, как они достигнут моделей, чувствительных к нагрузке. Благодаря отсутствию тяжёлых зависимостей цикл приёма может масштабироваться независимо от циклов исполнения, что особенно важно при пиковых нагрузках на платформу в начале учебных занятий, при массовых отправках решений студентами и в периоды контрольных мероприятий.

В фазе исполнения (Service loop) сервис-менеджер извлекает сообщение из очереди соответствующего сервиса, выполняет преобразование входных полей в соответствии с описанием шага цепочки, валидирует запрос по строгой схеме конкретного пайплайна, выполняет инференс модели и затем преобразование выходных полей в артефакты. Если цепочка содержит следующий шаг, сообщение помещается в очередь следующего сервиса, если шаг последний, ответ собирается по контракту команды и отправляется в очередь уведомлений платформы Мирера. Каждый сервис-менеджер удерживает в памяти собственный пул моделей в состоянии готовности и подтверждает обработку очередного сообщения брокеру только после успешного завершения соответствующего шага, что обеспечивает атомарность доставки и устойчивость системы к сбоям отдельных компонентов: при аварийном завершении обработки сообщение возвращается в очередь и поднимается другим экземпляром сервис-менеджера.

2.2 LLMServiceManager как единая точка работы с моделями

Ранее в реестре маршрутизатора был зарегистрирован единственный класс сервис-менеджера – LLMServiceManager. Этот компонент реализовывал несколько обязанностей: управление пулом экземпляров языковых моделей, разрешение конфликтов одновременного доступа через механизм захвата и освобождения моделей, выбор конкретного пайплайна по имени и непосредственное взаимодействие с моделью через подсистему LLMManager. Пул моделей размещался в режиме горячего ожидания: при старте сервиса все используемые модели загружались в видеопамять графических ускорителей, что устраняло задержки повторной инициализации, от десятков секунд до нескольких минут на одну модель в зависимости от её размера и метода загрузки, и обеспечивало низкую задержку отклика при поступлении запроса от платформы.

Помимо поддержки моделей, сервис-менеджер выполнял функции примитивного балансировщика нагрузки между экземплярами одной и той же модели: при поступлении запроса он искал свободный экземпляр среди занятых и, только при отсутствии такового, возвращал признак временной недоступности, побуждая брокер повторно поставить сообщение в очередь после небольшой задержки. В LLMServiceManager были также инкапсулированы логика опроса состояния моделей и сбор прикладных метрик – счётчиков обработанных запросов, гистограмм времени отклика, отметок ошибок инференса.

Для каждой образовательной задачи определялся отдельный пайплайн – самодостаточный модуль, содержащий строгую схему запроса и ответа, шаблон формирующей запрос инструкции и логику выполнения. Точкой входа в пайплайн служила асинхронная функция, принимающая валидированный запрос, экземпляр модели и ссылку на сервис-менеджер, возвращаемое значение представляло собой словарь именованных артефактов. Поиск пайплайнов был организован двухуровневым: сначала проверялась частная папка с реализациями, специфичными для платформы Мирера, затем – общая папка с обобщёнными пайплайнами. Такая организация позволяла переиспользовать общие пайплайны и при необходимости перекрывать их частными версиями без изменений в ядре маршрутизатора, что упрощало развитие функциональности и обеспечивало преемственность между поколениями реализаций.

2.3 Локальная обработка вложений

Образовательные задачи в платформе Мирера предполагают работу с разнообразными прикрепленными материалами: текстами условий заданий и решений студентов в форматах PDF и DOCX, презентациями преподавателей в PPTX, изображениями с рукописными решениями математических и инженерных задач, скан-копиями конспектов. В исходной архитектуре обработка этих вложений выполнялась локально внутри агента: компонент ContentProcessor объединял несколько библиотек парсинга документов и предоставлял пайплайнам уже подготовленный текст в унифицированном виде. Центральное место среди них занимала библиотека Docling [8] – специализированная система IBM, сочетающая нейросетевую модель сегментации макета страницы PP-DocLayoutV3 и сетевой компонент извлечения структурированного текста, способная обрабатывать сложные документы с таблицами, формулами и многоколоночной разметкой.

Подобный подход обеспечивал простоту первоначальной реализации и единый интерфейс для пайплайнов, но быстро выявил ряд недостатков. Дело в том, что сетевая составляющая библиотеки Docling требовала поддержки графического ускорителя, что существенно увеличивало время инициализации агента – на стандартной конфигурации старт занимал несколько десятков секунд из-за необходимости загрузки моделей сегментации в видеопамять. Это снижало гибкость развёртывания: даже в сценариях, в которых обработка вложений не предполагалась, агент по-прежнему резервировал ресурсы графического ускорителя для библиотеки распознавания. Любая модификация или обновление парсера требовали полной пересборки и перезапуска монолита, тогда как смена версии языковой модели в общем случае не должна была затрагивать обработку вложений и наоборот. Наконец, было невозможно независимо масштабировать ресурсы под задачи распознавания и под инференс языковых моделей: при росте потока вложений приходилось добавлять графические ускорители именно агенту, при том что нагрузки на языковые модели в это время могло не быть, и наоборот.

2.4 Ограничения исходной архитектуры

Анализ эксплуатации описанной архитектуры позволил выделить три ключевых ограничения, препятствующих подключению моделей иной модальности и развитию системы

в целом.

Во-первых, цепочки обработки и формирование ответа жёстко предполагали, что в конце пайплайна стоит языковая модель, а её выходное сообщение – текст определённой структуры. Для каждой команды в коде маршрутизатора присутствовала специфичная обёртка, извлекавшая отдельные поля результата работы LLM и собиравшая из них ответ платформе. Это нарушало принцип разделения обязанностей: код ядра маршрутизатора оказывался зависим от деталей конкретных пайплайнов, и любое изменение формата ответа конкретной образовательной задачи приводило к необходимости править ядро. Модель иной модальности, например, система оптического распознавания, возвращающая разметку документа, потребовала бы добавления собственного варианта обёртки и условной логики выбора способа сборки ответа в зависимости от типа сервиса, что лишь усугубило бы существующую связанность кода.

Во-вторых, в реестре сервис-фабрик был зарегистрирован единственный сервис, и сам интерфейс между маршрутизатором и сервис-менеджером не был отделён от специфики языковых моделей, в его сигнатурах фигурировали понятия, осмысленные только для LLM: параметры генерации, длина контекстного окна, обработка системного промпта. Добавление нового типа сервиса потребовало бы либо повторения этой специфики в новом классе с фиктивными значениями полей, не имеющих смысла для моделей иной модальности, либо более глубокой переработки маршрутизатора и сопутствующих интерфейсов с распространением изменений на все существующие пайплайны.

В-третьих, инкорпорация обработки документов в код самого агента создавала жёсткую связь между планированием ресурсов под нейросетевую модуль и под распознавание документов, описанную в разделе 2.3. К этому добавлялась организационная сложность: обновление компонентов распознавания требовало координации между разработчиками нейросетевого модуля и специалистами, поддерживающими инфраструктуру распознавания, при том, что эти подсистемы концептуально решают независимые задачи.

Совокупность перечисленных ограничений делала развитие платформы в сторону мультимодальной обработки запросов нецелесообразным без существенной архитектурной переработки. В следующем разделе описано, какие принципы были положены в основу новой архитектуры, устраняющей эти ограничения.

3. Универсальная мультимодальная архитектура

3.1 Принцип разделения маршрутизации и работы с моделью

В основу переработанной архитектуры положен принцип строгого разделения обязанностей между маршрутизатором запросов и сервис-менеджерами конкретных моделей. Маршрутизатор отвечает только за приём сообщений, валидацию их структуры, маршрутизацию по очередям сервисов и сборку ответа по контракту команды, он не содержит информации о том, какие именно типы моделей подключены к платформе и каким образом устроена работа с ними. Сервис-менеджеры, в свою очередь, инкапсулируют всю специфику работы с моделями своего типа: от управления набором экземпляров до особенностей форматирования входных и выходных данных.

Связующим элементом этих компонентов является класс `BaseServiceManager`, определяющий единый контракт сервис-менеджера через четыре обязательных асинхронных метода: метод поиска свободного экземпляра модели для указанного пайплайна, метод резервирования модели для эксклюзивной обработки запроса, метод освобождения модели после завершения обработки, метод выполнения полной цепочки «препроцессинг, инференс, постпроцессинг» в рамках конкретного пайплайна. Дополнительно класс предоставляет реализацию метода сбора прикладных метрик по умолчанию, что обеспечивает однородный мониторинг любого подключённого сервиса без необходимости дублирования телеметрической инфраструктуры в каждом производном классе. Сигнатуры методов сформулированы в терминах, нейтральных по отношению к модальности модели: вместо специфичных для языковых моделей параметров (длина контекстного окна, параметры генерации, обработка системного промпта) в них фигурируют обобщённые понятия – идентификатор пайплайна, словарь полей запроса, словарь именованных артефактов на выходе.

Подключение нового типа модели сводится к реализации этого контракта в производном классе и регистрации фабрики сервиса в реестре маршрутизатора. Никакие изменения в логике приёма запросов, маршрутизации, обработки ошибок или сборки ответов при этом не требуются. Реестр сервис-фабрик представляет собой словарь, отображающий имя сервиса в соответствующий класс, маршрутизатор при старте инициализирует по

экземпляру каждого зарегистрированного сервиса и запускает для них отдельные циклы-потребители сообщений на соответствующих очередях. Дополнительный практический эффект подобного разделения – упрощение модульного тестирования: каждый сервис-менеджер может проверяться в изоляции от остальных компонентов системы с использованием макетных реализаций моделей.

3.2 Конфигурируемые цепочки обработки

Описание того, через какие сервисы и пайплайны проходит конкретная команда, вынесено из кода в конфигурационные файлы. Для каждой команды цифровой образовательной платформы определены два файла: описание цепочки шагов и контракт взаимодействия с платформой Мирера.

Файл цепочки задаёт последовательность шагов с явным указанием сервиса, имени пайплайна и правил преобразования полей между соседними шагами. Преобразование может включать переименование переменных, чтение значений из объектного хранилища S3 по указанному пути и обратное сохранение результатов в S3 – это особенно важно для крупных артефактов (текстов транскрипций, образов документов, аудиофайлов), передавать которые целиком в сообщениях нецелесообразно из-за ограничений брокера на размер сообщения. Пути в хранилище задаются относительно общего префикса, привязанного к идентификатору запроса, что обеспечивает изоляцию данных разных запросов друг от друга и упрощает их утилизацию после завершения обработки. Пример сложной цепочки – команда транскрипции и резюмирования аудиозаписи занятия: первый шаг обрабатывается в сервисе распознавания речи и формирует выходное поле с транскрибированным текстом, второй шаг – в сервисе языковых моделей с пайплайном суммаризации, для которого это поле автоматически переименовывается во входное и подаётся как контекст для генерации обобщения.

Файл контракта описывает, какие поля платформа Мирера обязана передать при запросе и каким образом из артефактов цепочки собирается успешный ответ либо ответ об ошибке. Маршрутизатор не содержит зашитых в код шаблонов ответов: вся семантика выходного сообщения определяется контрактом, что позволяет добавлять новые команды и менять формат ответа без модификации кода ядра. Это особенно важно для согласования версий протокола с платформой Мирера:

эволюция формата сообщений происходит на уровне конфигурации и допускает одновременную поддержку нескольких версий контракта одной и той же команды, что критически важно для обеспечения совместимости при поэтапном развёртывании обновлений.

3.3 Унифицированный поток выходных артефактов

Одной из ключевых переработок при переходе к универсальной архитектуре стало устранение специфичных для каждого пайплайна обёрток выходного потока. В исходной реализации каждый пайплайн самостоятельно формировал структуру выходного сообщения для платформы Мирера, что приводило к дублированию логики сборки ответа: одни и те же преобразования полей повторялись в коде разных пайплайнов, а маршрутизатор оказывался привязан к деталям именования и формата результатов конкретных пайплайнов. При появлении нового пайплайна разработчику требовалось не только реализовать собственно логику обработки, но и продублировать инфраструктурный код сборки сообщения.

В новой архитектуре любой сервис, будь то языковая модель, система оптического распознавания или перспективные сервисы распознавания и синтеза речи, возвращает результат в виде единообразного словаря с именованными полями (артефактами). Преобразование этих артефактов в сообщение для следующего шага цепочки или в финальный ответ платформе выполняется единым компонентом – конфигурируемым трансформером сообщений, который читает правила преобразования из описания цепочки. Трансформер поддерживает переименование полей, чтение и запись артефактов в объектное хранилище S3, простые преобразования типов и подстановку значений по умолчанию. Это обеспечивает однотипность поведения маршрутизатора для всех типов сервисов и позволяет, в частности, использовать в одной цепочке артефакты как от пайплайна языковой модели, так и от пайплайна распознавания изображений, не прибегая к специальным склейкам кода. Дополнительный практический эффект – упрощение модульного тестирования пайплайнов: их корректность проверяется по соответствию словаря возвращаемых артефактов схеме, без необходимости в каждом тесте конструировать целевой формат выходного сообщения для конкретной команды.

3.4 Пайплайн как самодостаточный модуль

Каждый пайплайн оформляется в виде отдельной директории, содержащей минимально необходимый набор файлов: строгую схему запроса и ответа на основе декларативного описания структур данных, реализацию логики выполнения, шаблоны формирующих запрос инструкций (для пайплайнов языковых моделей) или конфигурацию обращений к сервису (для пайплайнов иной модальности). Точкой входа в пайплайн служит асинхронная функция, принимающая валидированный запрос, экземпляр модели и ссылку на сервис-менеджер, возвращаемое значение представляет собой словарь именованных артефактов, который в дальнейшем обрабатывается трансформером сообщений в соответствии с описанием шага цепочки. Использование строгой типизации схем входа и выхода исключает возможность неявных ошибок интеграции между шагами и обеспечивает быстрое обнаружение несоответствий ещё на этапе валидации запроса.

Подобная организация поддерживает иерархический поиск пайплайнов «частное – общее»: при наличии в системе реализации, специфичной для платформы Мирера, она имеет приоритет, при её отсутствии используется обобщённая реализация из общего каталога. Это позволяет переиспользовать общие пайплайны (например, базовый пайплайн транскрипции аудиозаписи) в специфичных для Мирера цепочках без копирования кода и одновременно изолировать частную бизнес-логику – особенности форматирования заданий для конкретных курсов, специфичные ограничения на длину ответа и дополнительные постфильтры в рамках частных пайплайнов, не загромождая ими общий каталог.

С организационной точки зрения такая структура существенно упрощает развитие функциональности: добавление новой образовательной задачи сводится к созданию новой директории пайплайна и регистрации соответствующей команды в реестре цепочек и контрактов. Никакие изменения в коде маршрутизатора или сервис-менеджеров для этого не требуются, что снижает риск нежелательных побочных эффектов при релизе новой функциональности и сокращает срок ввода новых образовательных сценариев в эксплуатацию.

4. Дополнительные особенности новой архитектуры

4.1 Поддержка специализированных типов вложений

Унификация работы с вложениями в рамках компонента `ContentProcessor` сделала возможным быстрое добавление поддержки новых типов учебных материалов. В частности, реализована обработка вложений в формате `LaTeX`, что особенно важно для образовательных курсов математического, физического и инженерного профиля, где условия задач, решения студентов и методические материалы преподавателей зачастую записываются на этом языке разметки. Поскольку файл `LaTeX` представляет собой структурированный текстовый документ с фиксированной кодировкой, его обработка не требует обращения к подсервису распознавания: содержимое декодируется сразу и передаётся в пайплайн напрямую. Аналогичный быстрый путь предусмотрен и для простых текстовых вложений с расширениями `.txt` и т.д.

Интерфейс работы со всеми типами вложений остаётся единым: пайплайн получает уже подготовленное текстовое представление документа независимо от того, был ли исходный файл обработан удалённым сабсервисом распознавания, локальным резервным парсером или прямым декодированием. Это упрощает написание пайплайнов, способных принимать материалы в произвольных форматах, и снимает с разработчика пайплайна необходимость учитывать особенности каждого формата отдельно. Добавление поддержки нового типа вложений сводится к описанию способа извлечения текста и регистрации новой ветви в карте обработчиков расширений в составе `ContentProcessor`, что упрощает развитие системы по мере появления новых типов учебных материалов, например, исходного кода в архивах различных языков программирования, формы интерактивных заданий или вложений со специализированной разметкой.

4.2 Классификация и обработка ошибок

В исходной архитектуре каждый пайплайн самостоятельно отвечал за перехват и интерпретацию ошибок инференса, что приводило к дублированию кода и неоднородной семантике ответов: одни и те же по природе ошибки могли быть классифицированы по-разному в разных пайплайнах, что усложняло наблюдаемость и поддержку системы. В новой архитектуре принят принцип «пайплайны без перехвата исключений»: возникающие в ходе

исполнения исключения прокидываются до маршрутизатора, который классифицирует их через специальный компонент-обработчик ошибок и принимает решение о реакции системы.

Ошибки делятся на две основные категории. Транзиентные ошибки (превышение времени ожидания при обращении к модели, потеря соединения с моделью, временная недоступность сабсервиса, отсутствие свободного экземпляра модели в пуле) приводят к возврату сообщения в очередь для повторной обработки – с учётом ограничения общего числа попыток и нарастающей задержки между ними. Постоянные ошибки (нарушение схемы запроса, отсутствие требуемого пайплайна, фатальная ошибка инференса, неверная конфигурация цепочки) переводят запрос в состояние неуспешной обработки: маршрутизатор формирует ответ об ошибке по контракту команды, обогащая его кодом из каталога постоянных ошибок. Каталог постоянных ошибок ведётся как часть конфигурации и описывает каждую категорию ошибки с её кодом, кратким описанием для целей логирования и пользовательским сообщением, отображаемым в платформе Мирера. Подобная классификация унифицирована для всех типов сервисов: сервис языковых моделей, сабсервис распознавания и перспективные сервисы распознавания и синтеза речи используют один и тот же механизм категоризации, что обеспечивает однородность реакции системы на ошибочные ситуации независимо от модальности обрабатываемых данных и существенно упрощает работу команды эксплуатации по диагностике инцидентов.

4.3. Единые метрики и наблюдаемость

Реализация метода сбора метрик непосредственно в базовом классе `BaseServiceManager` обеспечивает однородный сбор телеметрических данных любого подключаемого сервиса. Метрики экспортируются в систему мониторинга Prometheus [9] через стандартный эндпоинт публикации и включают: счётчики принятых и завершённых запросов с разбивкой по пайплайнам, гистограммы времени обработки с настраиваемыми границами интервалов для гранулярного анализа задержек, индикаторы текущего состояния пула моделей (число занятых и свободных экземпляров каждой модели), счётчики ошибок с категоризацией по типу. На стороне сервиса перечисленные метрики обновляются автоматически в рамках стандартных операций

сервис-менеджера и не требуют от разработчика конкретного производного класса какого-либо дополнительного кода.

Поскольку набор метрик и их семантика идентичны для всех сервисов, мониторинг и алертинг настраиваются однократно и распространяются на любые будущие сервисы – это снимает с разработчика нового сервис-менеджера задачу проектирования собственной телеметрии и существенно упрощает работу команды эксплуатации: однотипные информационные панели Grafana используются для всех сервисов, а пороговые значения алертов формулируются в терминах, не зависящих от типа модели. Дополнительный практический эффект – возможность сквозного отслеживания обработки запроса по всей цепочке шагов, в том числе пересекающих границы сервисов: единые идентификаторы запроса и согласованные семантики временных метрик позволяют выстраивать целостные временные диаграммы обработки и быстро локализовывать узкие места в составных цепочках обработки.

4.4 Готовность архитектуры к подключению сервисов речи

В рамках переработки заложены каркасы для будущих сервисов распознавания и синтеза речи, потребность в которых обусловлена развитием в платформе Мирера лекционных и семинарских форматов обучения, а также параллельной работой над цифровым двойником преподавателя [4], уже использующим компоненты распознавания и синтеза речи в составе самостоятельной системы. В реестре сервис-фабрик предусмотрены соответствующие точки подключения для сервисов распознавания (ASR) и синтеза (TTS), в конфигурации цепочек обработки уже описаны примеры сложных мультимодальных цепочек, в частности, цепочка «транскрипция аудиозаписи занятия - суммаризация полученного текста языковой моделью» и более сложная цепочка «транскрипция - суммаризация - синтез речи на основе обобщённого текста», объединяющая три сервиса разных модальностей в единой последовательности обработки. Соответствующие правила преобразования полей между шагами и работы с объектным хранилищем для крупных аудио и текстовых артефактов уже сформулированы, согласованы и протестированы на макетных реализациях. На текущем этапе сервис распознавания речи и сервис синтеза речи реализованы в виде интерфейсных заглушек, корректно выполняющих контракт BaseServiceManager для целей интеграционного тестирования и проверки целостности цепочек, но возвращающих маркер нереализованности

при попытке фактической обработки запроса. После замены заглушек на полноценные реализации соответствующих моделей подключение нового типа сервиса произойдёт без изменений в коде маршрутизатора и сервис-менеджеров других модальностей, что и является основным практическим результатом проведённой архитектурной эволюции и подтверждением жизнеспособности предложенного подхода.

5. Заключение

В работе описана эволюция архитектуры нейросетевого модуля платформы Мирера, обеспечивающая универсальную поддержку мультимодальных сервисов ИИ. Показано, что исходное LLM-ориентированное ядро имело принципиальные ограничения.

Предложенная архитектура устраняет эти ограничения за счёт взаимосвязанных решений: абстракции с единым контрактом для сервисов любой модальности, вынесения цепочек обработки и контракта взаимодействия с платформой в конфигурационные файлы, унификации потока выходных артефактов с их преобразованием через конфигурируемый трансформер.

В дальнейшем планируется расширение поддерживаемых модальностей и набора сценариев с новыми моделями, а также улучшение текущей архитектуры и ее оптимизация.

Работа выполнена в рамках государственного задания НИЦ «Курчатовский институт» - НИИСИ по теме № FNEF-2024-0001 (1023032100070-3-1.2. 1).

Evolution of the neural network module architecture: supporting multimodal services in the Mirera digital educational platform

A. S. Karavaeva¹, A. M. Kulikov², N. S. Martynov³, K. A. Mashchenko⁴,
K. K. Pchelin⁵, E. D. Tarasuk⁶

¹National Research Center «Kurchatov Institute»- SRISA, Moscow, Russia, aleksandrakaravaeva@yandex.ru;

²National Research Center «Kurchatov Institute»- SRISA, Moscow, Russia, kulikrch@yandex.ru;

³National Research Center «Kurchatov Institute» - SRISA, Moscow, Russia, nikolai.martynov@math.msu.ru;

⁴National Research Center «Kurchatov Institute» - SRISA, Moscow, Russia; Lomonosov Moscow State University, Moscow, Russia;
State University of Management, Moscow, Russia, kirill.mashchenko@niisi.ru;

⁵National Research Center «Kurchatov Institute» - SRISA, Moscow, Russia, pcelinkosta0@gmail.com;

⁶National Research Center «Kurchatov Institute» - SRISA, Moscow, Russia, ekaterina.tarasuk@math.msu.ru;

Abstract. This paper examines the evolution of the neural network module architecture of the Mirera Digital Educational Platform in connection with the expanding range of applied artificial intelligence models. It is shown that the original LLM-oriented orchestrator core did not allow models of other modalities to be integrated without substantial reworking. A universal multimodal architecture is described, including a service manager abstraction based on a unified contract, configurable request-processing chains, and a unified stream of output artifacts. Examples of integrations are presented. It is shown that the proposed architecture enables the integration of new model types without modifying the router core and provides a basis for the further development of multimodal educational scenarios.

Keywords: digital educational platform, DEP Mirera, neural network orchestrator, multimodal models, microservice architecture, LLM, OCR, RabbitMQ, AI agents, educational technologies

Литература

1. Chu Z., Wang S., Xie J., Zhu T., Yan Y., Ye J., Zhong A., Hu X., Liang J., Yu P. S., Wen Q. LLM agents for education: Advances and applications // arXiv preprint arXiv:2503.11733. - 2025. URL: <https://arxiv.org/pdf/2503.11733> (дата обращения: 02.07.2026).
2. Платформа Мирера [Электронный ресурс]. - URL: <https://www.mirera.ru> (дата обращения: 02.07.2026).
3. Леонов А. Г., Мартынов Н. С., Машенко К. А., Паремузов М. С., Пчелин К. К., Шляхов А. В. Области применения больших языковых моделей для цифровых образовательных платформ // Труды НИИСИ. - 2025. - Т. 15, № 2. - С. 9-15. URL: <https://www.t-niisi.ru/jour/article/view/72/120> (дата обращения: 02.07.2026).
4. Кушниренко А. Г., Леонов А. Г., Машенко К. А., Мартынов Н. С., Пчелин К. К., Шляхов А. В. Интеграция нейросетевого ассистента обучающихся по программированию в цифровую образовательную платформу // Информатика и образование. - 2026. - Т. 41, № 1. - С. 23-36. URL: <https://info.infojournal.ru/jour/article/view/1426/783> (дата обращения: 02.07.2026).
5. Леонов А. Г., Машенко К. А., Мартынов Н. С., Пчелин К. К., Шляхов А. В. Автоматизированное формирование образовательных видеоматериалов на основе модульной системы цифрового двойника преподавателя // Программные продукты и системы. - 2025. - Т. 38, № 4. - С. 694-703. URL: <https://swsys.ru/files/2025-4/694-703.pdf> (дата обращения: 02.07.2026).
6. Гриднев Н. В., Караваева А. С., Леонов А. Г., Машенко К. А., Пчелин К. К., Тарасюк Е. Д. Гибридная модель обучения: новации в учёте посещаемости в цифровой образовательной платформе Мирера // Труды НИИСИ. - 2025. - Т. 15, № 1. - С. 58-64. URL: <https://www.t-niisi.ru/jour/article/view/10/20> (дата обращения: 02.07.2026).
7. Vinoski S. Advanced Message Queuing Protocol // IEEE Internet Computing. - 2006. - Vol. 10, No. 6. - P. 87-89. - DOI: 10.1109/MIC.2006.116.
8. Auer C., Lysak M., Nassar A., Dolfi M., Livathinos N., Vagenas P., Berrospi C., Omenetti M.,

Lindlbauer F., Dinkla K., Mishra L., Kim Y., Gupta S., de Lima R. T., Weber V., Morin L., Meijer I., Kuropiatnyk V., Staar P. W. Docling Technical Report // arXiv preprint arXiv:2408.09869. - 2024. URL: <https://arxiv.org/pdf/2408.09869> (дата обращения: 02.07.2026).

9. Brazil B. Prometheus: Up & Running: Infrastructure and Application Performance Monitoring. - 2nd ed. - O'Reilly Media, 2024. - 506 p.

Поступила в редакцию / Received: 25.06.2026.

Поступила после рецензирования / Revised: 30.06.2026.

Принята к печати / Accepted: 03.07.2026.

UDC 004.8:37.018.46

Evolution of the neural network module architecture: supporting multimodal services in the Mirera digital educational platform

A. S. Karavaeva¹, A. M. Kulikov², N. S. Martynov³, K. A. Mashchenko⁴,
K. K. Pchelin⁵, E. D. Tarasuk⁶

¹National Research Center «Kurchatov Institute» - SRISA, Moscow, Russia, aleksandrakaravaeva@yandex.ru;

²National Research Center «Kurchatov Institute»- SRISA, Moscow, Russia, kulikrch@yandex.ru;

³National Research Center «Kurchatov Institute» - SRISA, Moscow, Russia, nikolai.martynov@math.msu.ru;

⁴National Research Center «Kurchatov Institute» - SRISA, Moscow, Russia; Lomonosov Moscow State University, Moscow, Russia;
State University of Management, Moscow, Russia, kirill.mashchenko@niisi.ru;

⁵National Research Center «Kurchatov Institute» - SRISA, Moscow, Russia, pcelinkosta0@gmail.com;

⁶National Research Center «Kurchatov Institute»- SRISA, Moscow, Russia, ekaterina.tarasuk@math.msu.ru;

Abstract. This paper examines the evolution of the neural network module architecture of the Mirera Digital Educational Platform in connection with the expanding range of applied artificial intelligence models. It is shown that the original LLM-oriented orchestrator core did not allow models of other modalities to be integrated without substantial reworking. A universal multimodal architecture is described, including a service manager abstraction based on a unified contract, configurable request-processing chains, and a unified stream of output artifacts. Examples of integrations are presented. It is shown that the proposed architecture enables the integration of new model types without modifying the router core and provides a basis for the further development of multimodal educational scenarios.

Keywords: digital educational platform, DEP Mirera, neural network orchestrator, multimodal models, microservice architecture, LLM, OCR, RabbitMQ, AI agents, educational technologies

1. Introduction

In recent years, large language models (LLMs) have become one of the key areas of practical implementation of artificial intelligence in education. The emergence of high-performance open architectures, their availability as ready-to-use API services, and the reduction of hardware infrastructure requirements through quantization methods have made it possible to move from isolated experiments to the systematic integration of language models into digital educational platforms. Today, LLMs are used for automated generation of learning assignments and test questions, review of learners' solutions, development of interactive assistants for specific disciplines, personalization of educational trajectories, and reduction of teachers' administrative workload [1]. At the same time, the range of associated artificial intelligence models is also expanding: optical recognition, image processing, speech recognition, and speech synthesis systems are being connected to language models, forming tools for working with multimodal scenarios in digital educational platforms.

The Mirera Digital Educational Platform [2] has accumulated experience in integrating artificial intelligence models into a wide range of educational tasks. In [3], the criteria for selecting large language models for educational scenarios are summarized, a comparative description of the architectures used in the platform is provided, and specific scenarios for their use are described: from generating assignments and test variants to building student and teacher assistants. In [4], the experience of integrating a neural network assistant for programming learners is presented; this assistant is capable of analyzing students' solutions in real time and generating instructional recommendations directly within the platform environment. In [5], a system for the automated generation of educational video materials based on a modular digital teacher twin is presented; it combines automatic speech recognition (ASR), text-to-speech synthesis (TTS), and lip synchronization (LipSync) modules and demonstrates a significant reduction in the teacher's labor costs for preparing educational video materials while maintaining their high quality. In addition to neural network services proper, the platform also develops related areas of educational

process automation, in particular hybrid attendance tracking with QR verification and video identification [6], which together form an integrated infrastructure for intelligent support of participants in the educational process.

The educational process is multimodal by nature: in addition to textual formulations of tasks and solutions, it involves images and diagrams from educational materials, audio and video recordings of lectures, handwritten notes, and teachers' presentations. As educational scenarios in a digital platform become more complex, the range of applied artificial intelligence models naturally goes beyond purely language-based systems: optical recognition systems for educational materials and handwritten text, speech recognition and synthesis models, and other neural network modules become increasingly relevant. Each of these areas requires models with their own input-data modality, specific hardware configuration (for example, specialized networks for optical recognition and vision-language models for documents with complex structure), and their own semantics of the output result, such as structured text, an audio signal, or a graphical representation. Implementing such scenarios within a single educational platform requires that the neural network module core be capable of connecting models of different natures without substantial reworking of routing, platform interaction contracts, or request-processing infrastructure.

The implementation of the scenarios described in [3; 4] required the creation of an independent neural network module that interacts with the Mirera platform core through message queues and performs the functions of request routing, model-pool management, and response assembly according to the contract with the platform. Initially, this module was designed for a single class of tasks: processing text requests using LLMs. This determined its architectural feature: all request-processing chains rigidly assumed that a language model stood at the end of the workflow, while attachment processing (documents and images) was performed locally inside the agent using a recognition library, thereby introducing substantial hardware dependencies into the monolith.

With the emergence of tasks involving other modalities, the incompatibility of this architecture with an expanded range of models became apparent. Connecting each new type of service within this architecture would require either duplicating the routing infrastructure or making point modifications to the core for the specifics of a particular service; both approaches lead to increasing complexity and poorer maintainability. In addition, placing recognition

dependencies inside the monolith did not allow resources for recognition tasks and LLM inference to be scaled independently, although they have significantly different load profiles and hardware requirements. By LLM inference, we mean the stage at which an already trained LLM processes a user request and generates a response without changing its internal parameters.

This paper describes the architectural evolution of the neural network module of the Mirera platform toward universal support for multimodal services. Section 2 examines the original LLM-oriented architecture and its limitations. Section 3 describes the transition to a universal multimodal architecture with a unified service manager contract, configurable chains, and a unified stream of output artifacts. Section 4 presents practical implementations of the capabilities of the new architecture.

2. Original LLM-Oriented Architecture

2.1 Two-Phase Request Processing

The neural network module of the Mirera platform is based on asynchronous request processing through the RabbitMQ message broker, which uses the AMQP protocol [7]. Requests from the platform arrive as structured envelope messages in an intake queue; responses are placed in a notification queue and delivered back to the platform for displaying the result to participants in the educational process. Between these endpoints, request processing is divided into two phases, each implemented by its own message-consumer loop.

In the intake phase (Intake loop), the router validates the structure of the incoming envelope message, loads the description of the processing chain for the corresponding command from a configuration file, checks the payload for compliance with the command contract, and places the message in the queue of the first service in the chain. This phase does not perform computational work and provides strict input control: invalid requests with a malformed message structure, an unknown command name, or a payload that does not comply with the contract are rejected before they reach load-sensitive models. Due to the absence of heavy dependencies, the intake loop can be scaled independently of execution loops, which is especially important during peak platform loads, such as the beginning of classes, mass submission of student solutions, and assessment periods.

In the execution phase (Service loop), the service manager retrieves a message from the queue of the corresponding service, transforms the input fields according to the description of the chain step,

validates the request against the strict schema of a specific pipeline, performs model inference, and then transforms the output fields into artifacts. If the chain contains a next step, the message is placed in the queue of the next service; if the step is the last one, the response is assembled according to the command contract and sent to the notification queue of the Mirera platform. Each service manager keeps its own pool of models in a ready state in memory and acknowledges processing of the next message to the broker only after successful completion of the corresponding step. This ensures delivery atomicity and system resilience to failures of individual components: if processing terminates abnormally, the message is returned to the queue and picked up by another service-manager instance.

2.2 LLMServiceManager as a Single Point of Model Interaction

Previously, only one service-manager class, LLMServiceManager, was registered in the router registry. This component performed several responsibilities: managing a pool of language-model instances, resolving concurrent access conflicts through a model acquisition and release mechanism, selecting a specific pipeline by name, and directly interacting with the model through the LLManager subsystem. The model pool was maintained in a hot-standby mode: when the service started, all models used by it were loaded into the video memory of graphics accelerators. This eliminated reinitialization delays, ranging from tens of seconds to several minutes per model depending on its size and loading method, and provided low response latency when a request arrived from the platform.

In addition to model support, the service manager performed the functions of a primitive load balancer between instances of the same model: when a request arrived, it searched for a free instance among the busy ones and, only if none was available, returned a temporary-unavailability flag prompting the broker to requeue the message after a short delay. LLMServiceManager also encapsulated logic for polling model state and collecting application metrics, including counters of processed requests, response-time histograms, and inference-error markers.

A separate pipeline was defined for each educational task: a self-contained module comprising a strict request and response schema, a prompt template, and execution logic. The pipeline entry point was an asynchronous function that accepted a validated request, a model instance, and a reference to the service manager; the returned value was a dictionary of named artifacts. Pipeline lookup was organized in two levels: first, a private folder with implementations specific to the Mirera

platform was checked, and then a common folder with generalized pipelines was checked. This organization made it possible to reuse common pipelines and, when necessary, override them with private versions without changing the router core, thereby simplifying functional development and ensuring continuity between generations of implementations.

2.3 Local Processing of Attachments

Educational tasks in the Mirera platform involve working with a variety of attached materials: texts of assignment statements and student solutions in PDF and DOCX formats, teachers' presentations in PPTX, images containing handwritten solutions to mathematical and engineering problems, and scanned copies of notes. In the original architecture, these attachments were processed locally inside the agent: the ContentProcessor component combined several document-parsing libraries and provided pipelines with already prepared text in a unified form. A central role among these libraries was played by Docling [8], a specialized IBM system combining the PP-DocLayoutV3 neural network page-layout segmentation model and a network component for extracting structured text; it is capable of processing complex documents with tables, formulas, and multi-column layouts.

This approach ensured the simplicity of the initial implementation and a unified interface for pipelines, but it quickly revealed a number of disadvantages. The network component of the Docling library required graphics accelerator support, which significantly increased the agent initialization time: in a standard configuration, startup took several tens of seconds because segmentation models had to be loaded into video memory. This reduced deployment flexibility: even in scenarios in which attachment processing was not expected, the agent still reserved graphics-accelerator resources for the recognition library. Any modification or update of the parser required a full rebuild and restart of the monolith, whereas changing the version of a language model should generally not affect attachment processing, and vice versa. Finally, it was impossible to scale resources independently for recognition tasks and language-model inference: as the flow of attachments increased, graphics accelerators had to be added specifically to the agent, even though there might be no load on the language models at that time, and vice versa.

2.4 Limitations of the Original Architecture

An analysis of the operation of the described architecture made it possible to identify three key limitations that hindered the integration of models

of other modalities and the development of the system as a whole.

First, the processing chains and response generation rigidly assumed that a language model was placed at the end of the pipeline and that its output message was text of a certain structure. For each command, the router code contained a specific wrapper that extracted individual fields from the LLM result and assembled the response for the platform. This violated the principle of separation of concerns: the router core code became dependent on the details of specific pipelines, and any change in the response format of a specific educational task required modifications to the core. A model of another modality, for example an optical recognition system returning document markup, would require the addition of its own wrapper variant and conditional logic for selecting the response assembly method depending on the type of service, which would only further increase the existing code coupling.

Second, only one service was registered in the service-factory registry, and the interface between the router and the service manager was not separated from the specifics of language models: its signatures contained concepts meaningful only for LLMs, such as generation parameters, context-window length, and system-prompt handling. Adding a new service type would require either repeating this specificity in a new class with dummy values for fields that have no meaning for models of another modality, or performing a deeper reworking of the router and related interfaces with changes propagated to all existing pipelines.

Third, incorporating document processing into the code of the agent itself created a rigid coupling between resource planning for the neural network module and for document recognition, as described in Section 2.3. This was compounded by organizational complexity: updating recognition components required coordination between developers of the neural network module and specialists maintaining the recognition infrastructure, even though these subsystems conceptually solve independent tasks.

Taken together, these limitations made the development of the platform toward multimodal request processing impractical without substantial architectural reworking. The next section describes the principles underlying the new architecture that eliminates these limitations.

3. Universal Multimodal Architecture

3.1 Principle of Separating Routing from Model Operation

The redesigned architecture is based on the

principle of strict separation of responsibilities between the request router and the service managers of specific models. The router is responsible only for receiving messages, validating their structure, routing them through service queues, and assembling the response according to the command contract; it contains no information about which model types are connected to the platform or how they operate. Service managers, in turn, encapsulate all specifics of working with models of their type, from managing a set of instances to formatting input and output data.

The connecting element between these components is the `BaseServiceManager` class, which defines a unified service-manager contract through four mandatory asynchronous methods: a method for finding a free model instance for the specified pipeline, a method for reserving a model for exclusive request processing, a method for releasing a model after processing is complete, and a method for executing the full “preprocessing, inference, postprocessing” chain within a specific pipeline. In addition, the class provides a default implementation of the application-metrics collection method, which ensures uniform monitoring of any connected service without the need to duplicate telemetry infrastructure in each derived class. Method signatures are formulated in terms neutral with respect to the model modality: instead of parameters specific to language models (context-window length, generation parameters, system-prompt handling), they use generalized concepts: pipeline identifier, request-field dictionary, and dictionary of named output artifacts.

Connecting a new model type is reduced to implementing this contract in a derived class and registering the service factory in the router registry. No changes to request intake, routing, error handling, or response assembly logic are required. The service-factory registry is a dictionary that maps a service name to the corresponding class; at startup, the router initializes one instance of each registered service and launches separate message-consumer loops for them on the corresponding queues. An additional practical effect of this separation is simplified unit testing: each service manager can be tested in isolation from the other system components using mock model implementations.

3.2 Configurable Processing Chains

The description of which services and pipelines a particular command passes through is moved from code to configuration files. Two files are defined for each command of the digital educational platform: a step-chain description and a contract for interaction with the Mirera platform.

The chain file specifies a sequence of steps with explicit indication of the service, pipeline name, and

rules for transforming fields between adjacent steps. Transformation may include renaming variables, reading values from S3 object storage at a specified path, and saving results back to S3. This is especially important for large artifacts, such as transcript texts, document images, and audio files, which are impractical to transmit in messages in their entirety because of broker message-size limitations. Paths in storage are specified relative to a common prefix associated with the request identifier, which ensures isolation of data from different requests and simplifies its disposal after processing is complete. An example of a complex chain is a command for transcribing and summarizing an audio recording of a class: the first step is executed in the speech recognition service and produces an output field with the transcribed text; the second step is executed in the language-model service with a summarization pipeline, for which this field is automatically renamed as an input field and supplied as context for generating a summary.

The contract file describes which fields the Mirera platform must provide in a request and how a successful response or an error response is assembled from chain artifacts. The router contains no response templates hard-coded in the source code: the entire semantics of the output message is determined by the contract, which makes it possible to add new commands and change the response format without modifying the core code. This is especially important for coordinating protocol versions with the Mirera platform: the evolution of message formats occurs at the configuration level and allows simultaneous support for several contract versions of the same command, which is critical for maintaining compatibility during staged deployment of updates.

3.3 Unified Output Artifact Stream

One of the key reworkings in the transition to the universal architecture was the elimination of output-stream wrappers specific to each pipeline. In the original implementation, each pipeline independently formed the structure of the output message for the Mirera platform, which led to duplication of response-assembly logic: the same field transformations were repeated in the code of different pipelines, while the router became tied to naming details and result formats of specific pipelines. When a new pipeline appeared, the developer had to implement not only the processing logic itself but also duplicate the infrastructure code for assembling the message.

In the new architecture, any service, whether a language model, an optical recognition system, or prospective speech recognition and synthesis services, returns a result as a uniform dictionary with named fields (artifacts). Transformation of

these artifacts into a message for the next step of the chain or into the final response to the platform is performed by a single component, a configurable message transformer that reads transformation rules from the chain description. The transformer supports field renaming, reading and writing artifacts to S3 object storage, simple type conversions, and substitution of default values. This ensures uniform router behavior for all service types and, in particular, makes it possible to use artifacts from both a language-model pipeline and an image-recognition pipeline within a single chain without resorting to special code glue. An additional practical effect is simplified unit testing of pipelines: their correctness is checked by the conformity of the returned artifact dictionary to the schema, without the need to construct the target output-message format for a specific command in each test.

3.4 Pipeline as a Self-Contained Module

Each pipeline is organized as a separate directory containing the minimum necessary set of files: a strict request and response schema based on a declarative description of data structures, an implementation of the execution logic, templates for prompt-forming instructions (for language-model pipelines), or configuration for calls to the service (for pipelines of another modality). The pipeline entry point is an asynchronous function that accepts a validated request, a model instance, and a reference to the service manager; the returned value is a dictionary of named artifacts, which is subsequently processed by the message transformer according to the chain-step description. The use of strong typing for input and output schemas eliminates the possibility of implicit integration errors between steps and ensures rapid detection of inconsistencies already at the request validation stage.

This organization supports hierarchical pipeline lookup, “private - common”: if an implementation specific to the Mirera platform is present in the system, it takes priority; if it is absent, the generalized implementation from the common catalog is used. This makes it possible to reuse common pipelines (for example, a basic audio-recording transcription pipeline) in Mirera-specific chains without code duplication and at the same time to isolate private business logic, such as assignment-formatting features for specific courses, specific limits on response length, and additional postfilters within private pipelines, without cluttering the common catalog with them.

From an organizational point of view, this structure significantly simplifies functional development: adding a new educational task is reduced to creating a new pipeline directory and registering the corresponding command in the

registry of chains and contracts. No changes to the router or service-manager code are required, which reduces the risk of undesirable side effects when releasing new functionality and shortens the time needed to put new educational scenarios into operation.

4. Additional Features of the New Architecture

4.1 Support for Specialized Attachment Types

Unifying the handling of attachments within the ContentProcessor component made it possible to quickly add support for new types of educational materials. In particular, processing of attachments in LaTeX format has been implemented, which is especially important for mathematical, physical, and engineering courses, where task statements, student solutions, and teachers' methodological materials are often written in this markup language. Since a LaTeX file is a structured text document with a fixed encoding, its processing does not require a call to a recognition subservice: the content is decoded immediately and passed directly to the pipeline. A similar fast path is also provided for simple text attachments with extensions such as .txt and others.

The interface for working with all attachment types remains unified: the pipeline receives an already prepared textual representation of the document regardless of whether the original file was processed by a remote recognition subservice, a local fallback parser, or direct decoding. This simplifies the development of pipelines capable of accepting materials in arbitrary formats and relieves the pipeline developer of the need to account for the specifics of each format separately. Adding support for a new attachment type is reduced to describing the method for extracting text and registering a new branch in the extension-handler map within ContentProcessor. This simplifies system development as new types of educational materials appear, for example source code in archives for various programming languages, interactive-task forms, or attachments with specialized markup.

4.2 Error Classification and Handling

In the original architecture, each pipeline was independently responsible for intercepting and interpreting inference errors, which led to code duplication and nonuniform response semantics: errors of the same nature could be classified differently in different pipelines, complicating observability and system support. The new architecture adopts the principle of "pipelines without exception interception": exceptions arising during execution are propagated to the router, which

classifies them through a special error-handler component and decides how the system should respond.

Errors are divided into two main categories. Transient errors (timeout when accessing a model, loss of connection to a model, temporary unavailability of a subservice, or absence of a free model instance in the pool) cause the message to be returned to the queue for repeated processing, subject to a limit on the total number of attempts and an increasing delay between them. Permanent errors (violation of the request schema, absence of the required pipeline, fatal inference error, or incorrect chain configuration) move the request into a failed-processing state: the router forms an error response according to the command contract and enriches it with a code from the catalog of permanent errors. The catalog of permanent errors is maintained as part of the configuration and describes each error category with its code, a short description for logging purposes, and a user-facing message displayed in the Mirera platform. This classification is unified for all service types: the language-model service, the recognition subservice, and prospective speech recognition and synthesis services use the same categorization mechanism, which ensures a uniform system response to error situations regardless of the modality of the processed data and substantially simplifies incident diagnostics for the operations team.

4.3 Unified Metrics and Observability

Implementing the metrics collection method directly in the BaseServiceManager base class ensures uniform collection of telemetry data for any connected service. Metrics are exported to the Prometheus monitoring system [9] through a standard publication endpoint and include counters of received and completed requests broken down by pipeline, processing-time histograms with configurable bucket boundaries for granular latency analysis, indicators of the current state of the model pool (the number of busy and free instances of each model), and error counters categorized by type. On the service side, these metrics are updated automatically as part of standard service-manager operations and do not require any additional code from the developer of a specific derived class.

Because the set of metrics and their semantics are identical for all services, monitoring and alerting are configured once and then apply to any future services. This relieves the developer of a new service manager from the task of designing custom telemetry and significantly simplifies the work of the operations team: uniform Grafana dashboards are used for all services, and alert thresholds are formulated in terms independent of the model type. An additional practical effect is the ability to trace request processing end to end across the entire

sequence of steps, including steps that cross service boundaries: unified request identifiers and agreed semantics of timing metrics make it possible to build complete time diagrams of processing and quickly localize bottlenecks in composite processing chains.

4.4 Architecture Readiness for Connecting Speech Services

As part of the reworking, frameworks have been laid for future speech recognition and speech synthesis services. The need for them is driven by the development of lecture and seminar learning formats in the Mirera platform, as well as by parallel work on the digital teacher twin [4], which already uses speech recognition and synthesis components as part of an independent system. The service-factory registry provides the corresponding connection points for automatic speech recognition (ASR) and text-to-speech synthesis (TTS) services. The processing-chain configuration already describes examples of complex multimodal chains, in particular a “class audio recording transcription - summarization of the resulting text by a language model” chain and a more complex “transcription - summarization - speech synthesis based on the summarized text” chain, which combines three services of different modalities in a single processing sequence. The corresponding rules for field transformation between steps and for working with object storage for large audio and text artifacts have already been formulated, agreed upon, and tested on mock implementations.

At the current stage, the speech recognition service and the speech synthesis service are

implemented as interface stubs that correctly fulfill the BaseServiceManager contract for the purposes of integration testing and checking chain integrity, but return an unimplemented marker when actual request processing is attempted. After the stubs are replaced with full implementations of the corresponding models, connecting the new service type will occur without changes to the router code or to service managers of other modalities. This is the main practical result of the architectural evolution carried out and confirms the viability of the proposed approach.

5. Conclusion

This paper describes the evolution of the neural network module architecture of the Mirera platform, providing universal support for multimodal AI services. It is shown that the original LLM-oriented core had fundamental limitations.

The proposed architecture eliminates these limitations through a set of interrelated solutions: an abstraction with a unified contract for services of any modality, moving processing chains and the platform-interaction contract into configuration files, and unifying the output artifact stream with its transformation through a configurable transformer.

In the future, it is planned to expand the supported modalities and the set of scenarios involving new models, as well as to improve and optimize the current architecture.

The work was carried out within the framework of the state assignment of the National Research Center “Kurchatov Institute” - SRISA, topic No. FNEF-2024-0001 (1023032100070-3-1.2.1).

References

1. Chu Z., Wang S., Xie J., Zhu T., Yan Y., Ye J., Zhong A., Hu X., Liang J., Yu P. S., Wen Q. LLM agents for education: Advances and applications // arXiv preprint arXiv:2503.11733. - 2025. URL: <https://arxiv.org/pdf/2503.11733> (accessed: 02.07.2026).
2. Mirera Platform [Electronic resource]. - URL: <https://www.mirera.ru> (accessed: 02.07.2026).
3. Leonov A. G., Martynov N. S., Mashchenko K. A., Paremuzov M. S., Pchelin K. K., Shlyakhov A. V. Areas of application of large language models for digital educational platforms // Proceedings of SRISA. - 2025. - Vol. 15, No. 2. - P. 9-15. URL: <https://www.t-niisi.ru/jour/article/view/72/120> (accessed: 02.07.2026).
4. Kushnirenko A. G., Leonov A. G., Mashchenko K. A., Martynov N. S., Pchelin K. K., Shlyakhov A. V. Integration of a neural network assistant for programming learners into a digital educational platform // Informatics and Education. - 2026. - Vol. 41, No. 1. - P. 23-36. URL: <https://info.infojournal.ru/jour/article/view/1426/783> (accessed: 02.07.2026).
5. Leonov A. G., Mashchenko K. A., Martynov N. S., Pchelin K. K., Shlyakhov A. V. Automated generation of educational video materials based on a modular digital teacher twin system // Software & Systems. - 2025. - Vol. 38, No. 4. - P. 694-703. URL: <https://swsys.ru/files/2025-4/694-703.pdf> (accessed: 02.07.2026).
6. Gridnev N. V., Karavaeva A. S., Leonov A. G., Mashchenko K. A., Pchelin K. K., Tarasuk E. D.

Hybrid learning model: innovations in attendance tracking in the Mirera digital educational platform // Proceedings of SRISA. - 2025. - Vol. 15, No. 1. - P. 58-64. URL: <https://www.t-niisi.ru/jour/article/view/10/20> (accessed: 02.07.2026).

7. Vinoski S. Advanced Message Queuing Protocol // IEEE Internet Computing. - 2006. - Vol. 10, No. 6. - P. 87-89. - DOI: 10.1109/MIC.2006.116.

8. Auer C., Lysak M., Nassar A., Dolfi M., Livathinos N., Vagenas P., Berrospi C., Omenetti M., Lindlbauer F., Dinkla K., Mishra L., Kim Y., Gupta S., de Lima R. T., Weber V., Morin L., Meijer I., Kuropiatnyk V., Staar P. W. Docling Technical Report // arXiv preprint arXiv:2408.09869. - 2024. URL: <https://arxiv.org/pdf/2408.09869> (accessed: 02.07.2026).

9. Brazil B. Prometheus: Up & Running: Infrastructure and Application Performance Monitoring. - 2nd ed. - O'Reilly Media, 2024. - 506 p.

Поступила в редакцию / Received: 25.06.2026.

Поступила после рецензирования / Revised: 30.06.2026.

Принята к печати / Accepted: 03.07.2026.

УДК 004.8

Сравнительный анализ архитектур многоагентных дебатов крупных языковых моделей в задаче поддержки принятия решений

Ш. М. Магомедов¹

¹ Дагестанский государственный технический университет, Махачкала, Россия,
shakhmanay.magomedov@mail.ru

Аннотация. В статье представлен сравнительный анализ пяти архитектур многоагентных дебатов крупных языковых моделей в задаче поддержки принятия решений при оценке качества образовательных тестов. Цель исследования состояла в выявлении зависимости между топологией взаимодействия агентов и качеством решения, вычислительными затратами, временем ответа и устойчивостью результата, а также в определении условий, при которых применение многоагентных дебатов в системах поддержки принятия решений (СППР) является практически оправданным. На едином прикладном полигоне сопоставлялись базовая, линейная, циклическая, перекрестная архитектуры и архитектура с исключением модели. Корпус включал 60 тестов по четырем дисциплинам основного общего образования с разделением на корректные тесты и тесты с внесенными ошибками; ансамбль моделей был фиксирован и включал Qwen 2.5 72B Instruct, Grok 4.1 Fast и Gemma 4 26B-A4B IT. Для оценки использовались macro-F1, макро-средняя абсолютная ошибка, число токенов, стоимость на один тест и средняя длительность ответа. Установлено, что наибольшее наблюдаемое значение macro-F1 в рамках проведенного эксперимента показала архитектура с исключением модели, тогда как базовая архитектура оказалась лучшей по макро-средней абсолютной ошибке и сохранила сильные позиции по операционной эффективности. Дополнительно выявлено, что часть архитектур значительно чувствительна к типу теста, а увеличение числа раундов циклических дебатов после определенного порога не обеспечивает устойчивого прироста качества. Полученные результаты позволяют рассматривать архитектуру дебатов как управляемый параметр СППР, а не как нейтральную техническую деталь.

Ключевые слова: крупные языковые модели, многоагентные дебаты, поддержка принятия решений, образовательные тесты, языковая модель-судья, macro-F1, архитектура взаимодействия

1. Введение

Многоагентные дебаты крупных языковых моделей рассматриваются сегодня как один из путей повышения надежности рассуждений и оценок по сравнению с одиночным выводом модели [1, 5]. Работы по многоагентным дебатам показали, что коллективное обсуждение может улучшать фактическую корректность и качество рассуждений, однако более поздние исследования продемонстрировали, что результат чувствителен к самой схеме коммуникации между агентами и не определяется только их количеством [1, 2]. Дополнительную сложность вносит использование LLM в роли судьи: такие схемы масштабируемы, но потенциально уязвимы к систематическим смещениям, что особенно важно для задач поддержки решений [3, 4].

В образовательной диагностике это ограничение приобретает прикладной характер. Качество тестового задания определяется не только фактом наличия ошибок, но

и соответствием дисциплине, уровню обучающихся, корректностью эталонных ответов, ясностью формулировок и полнотой тематического покрытия [7–10]. Поэтому оценка таких объектов требует сопоставления архитектур дебатов на материале, где возможна эталонная разметка и где цена ложноположительных и ложноотрицательных решений различна. Образовательный контекст также уже используется в современных исследованиях LLM как самостоятельный класс многоагентных сценариев, что подтверждает методологическую уместность выбранного полигона [6].

Цель статьи – выполнить сравнительный анализ архитектур многоагентных дебатов крупных языковых моделей в задаче поддержки принятия решений при оценке качества образовательных тестов и определить, какие схемы взаимодействия дают более удачный баланс между качеством, затратами и устойчивостью.

В проведенном исследовании на едином

корпусе сопоставлялись пять архитектур: базовая, линейная, циклическая, перекрестная и архитектура с исключением модели. Корпус был организован как восемь файлов формата JSON по четырем дисциплинам основного общего образования Российской Федерации – история России, математика, информатика и русский язык – с разделением на корректные тесты и тесты с внесенными ошибками. Каждый тест содержал десять вопросов с четырьмя вариантами ответа, что дало 60 тестов и 600 заданий в сумме. Итоговая числовая оценка качества формировалась по пяти критериям C1–C5: соответствие дисциплине, соответствие уровню обучающихся, корректность эталонных ответов, ясность формулировок и полнота тематического покрытия; сами критерии и их веса задавались заранее на нормативной и методической основе [9, 10]. Фиксированный ансамбль включал Qwen 2.5 72B Instruct, Grok 4.1 Fast и Gemma 4 26B-A4B IT, а основными метриками выступали macro-F1 для классификации проблемных вопросов, макро-средняя абсолютная ошибка для числовой оценки качества теста, число токенов, стоимость на один тест и длительность ответа.

Тестовый корпус был сформирован автором как контролируемая экспериментальная выборка на основе содержания программ основного общего образования Российской Федерации для 7–9-х классов. Стандартные экзаменационные банки ОГЭ, ЕГЭ или международные наборы не использовались напрямую, поскольку для данной задачи требовалась парная структура «корректная версия / версия с внесенными ошибками», заранее известные проблемные вопросы и эталонные оценки по критериям C1–C5; такие поля в открытых экзаменационных наборах обычно отсутствуют. Дисциплины были

выбраны так, чтобы совместить формализованные задания (математика, информатика) и задания с более выраженным контекстным анализом (история России, русский язык). В качестве типовых объектов корпуса использовались, например, задания на вычисление или выбор корректного алгоритмического утверждения, а также вопросы на понимание исторического факта или языковой формулировки; в версии с внесенными ошибками отдельно закладывались неверные эталонные ответы, несоответствие уровню класса, неоднозначная формулировка и выход за пределы заявленной дисциплины. Доступ к моделям осуществлялся через API-агрегатор OpenRouter; все вызовы выполнялись с фиксированными параметрами генерации и сохранялись в структурированных JSON-логах.

Такой дизайн позволяет сравнивать не только точность, но и операционные свойства разных топологий дебатов на одном прикладном полигоне. Тем самым исследование направлено не на подтверждение общей полезности многоагентных дебатов, а на определение условий их рационального применения в СППР.

2. Результаты исследования

Первый этап анализа связан с проверкой основного предположения исследования: действительно ли усложнение архитектуры дебатов приводит к более качественному решению. Для этого архитектуры сопоставлялись не только по значению macro-F1, но и по макро-средней абсолютной ошибке, числу токенов, стоимости и времени ответа. В таблице 1 приведены сводные результаты сравнения архитектур многоагентных дебатов; для сопоставимости стоимостные показатели нормированы к одному тесту.

Таблица 1. Сводные показатели архитектур многоагентных дебатов

Архитектура	Macro-F1	Макро-средняя абсолютная ошибка	Токенов / тест	Стоимость / тест, USD	Длительность, с
Архитектура с исключением модели	0,772	0,099	24 155	0,0071	188,5
Базовая архитектура	0,701	0,052	15 208	0,0043	92,5
Циклическая архитектура	0,686	0,088	20 252	0,0054	140,7
Перекрестная архитектура	0,616	0,095	21 126	0,0058	165,0
Линейная архитектура	0,566	0,102	9 643	0,0027	94,0

Как видно из таблицы 1, наибольшее наблюдаемое значение макро-F1 получила архитектура с исключением модели – 0,772. Базовая архитектура показала 0,701, циклическая – 0,686, перекрестная – 0,616, линейная – 0,566. При этом по макро-средней абсолютной ошибке лучшей оказалась базовая архитектура – 0,052, тогда как у архитектуры с исключением модели эта метрика составила 0,099. Линейная архитектура потребовала наименьшего количества ресурсов – 9643 токена и 0,0027 USD на тест, – но уступила остальным по качеству решения. Следовательно, уже на уровне сводного сравнения видно, что архитектура дебатов влияет не только на точность, но и на тип компромисса между качеством и затратами. При этом различия

следует интерпретировать аккуратно: по данным попарных сравнений с критерием Вилкоксона и поправкой Холма статистически устойчивым в этой серии осталось только отличие между линейной архитектурой и архитектурой с исключением модели. Поэтому окончательное ранжирование остальных схем в рамках текущего корпуса носит прежде всего описательный характер.

Для оценки прикладной рациональности архитектур далее рассмотрим соотношение качества и стоимости. На рисунке 1 показано положение архитектур в пространстве «макро-F1 – стоимость на один тест».

Как видно из рисунка 1, на Парето-фронт попадают линейная, базовая архитектура и архитектура с исключением модели.

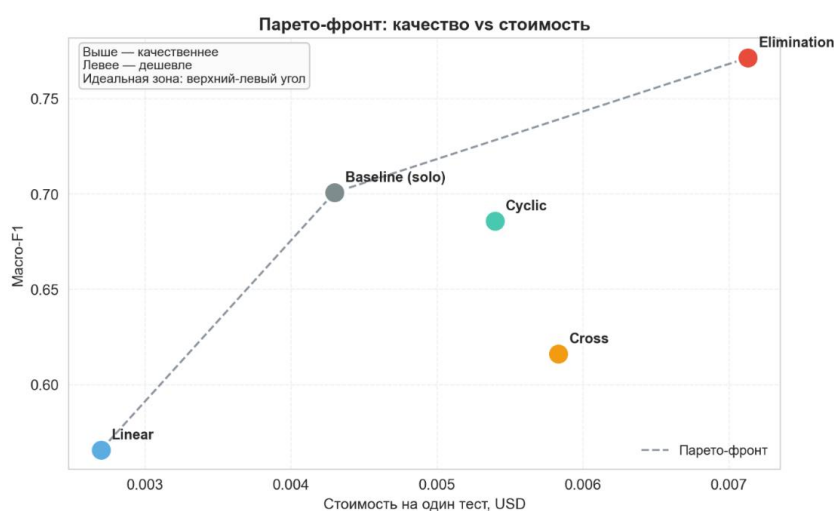


Рис. 1. Парето-сравнение архитектур в пространстве «макро-F1 – стоимость на тест»

Это означает, что при разных бюджетных режимах рациональными оказываются разные решения: линейная архитектура – при минимальных расходах, базовая – как умеренный и устойчивый компромисс, архитектура с исключением модели – при приоритете качества. Циклическая и перекрестная архитектуры в текущем эксперименте остаются вне Парето-фронта, поскольку требуют больше ресурсов без соответствующего выигрыша по макро-F1.

Практический вывод из этого состоит в том, что рост сложности взаимодействия между агентами не тождествен росту полезности архитектуры для СППР. Иными словами, более «плотная» коммуникация между моделями еще не означает более выгодное решение для реальной системы.

В таблице 2 приведены значения макро-F1 по двум срезам корпуса – корректные и с внесенными ошибками.

Таблица 2. Значения макро-F1 по типам тестов

Архитектура	Корректные	С внесенными ошибками	Δ , п.п.
Базовая архитектура	0,689	0,713	+2,4
Линейная архитектура	0,433	0,698	+26,5
Циклическая архитектура	0,567	0,805	+23,9
Перекрестная архитектура	0,433	0,799	+36,6
Архитектура с исключением модели	0,767	0,776	+1,0

Этот разрез важен потому, что одна и та же средняя метрика может скрывать различную чувствительность архитектур к ложным тревогам и пропуску реальных дефектов.

Как видно из таблицы 2, наиболее ровное поведение демонстрирует архитектура с исключением модели: 0,767 на корректных тестах и 0,776 на тестах с внесенными ошибками. Базовая архитектура также остается сравнительно устойчивой: 0,689 и 0,713 соответственно. Линейная, циклическая и перекрестная архитектуры ведут себя иначе: они заметно лучше работают на тестах с внесенными ошибками, чем на корректных тестах. Наиболее выраженная асимметрия наблюдается у перекрестной архитектуры – 0,433 на корректных против 0,799 на тестах с внесенными ошибками, то есть разрыв составляет 36,6 процентного пункта. Это позволяет заключить, что часть дебатных схем

подходит скорее для режима активного поиска дефектов, чем для режима осторожной проверки уже подготовленного банка заданий, где цена ложноположительного решения выше. Для практической СППР это критично: одна и та же архитектура может быть полезной в режиме «найти проблему любой ценой» и одновременно неудачной в режиме «не тревожить эксперта без достаточных оснований».

Отдельно был проверен вопрос о том, повышает ли увеличение числа раундов качество дебатов. На рисунке 2 изображена динамика макро-F1 и стоимости для циклической архитектуры при изменении числа раундов от 1 до 5.

Как видно из рис. 2, увеличение числа раундов не приводит к монотонному росту качества.

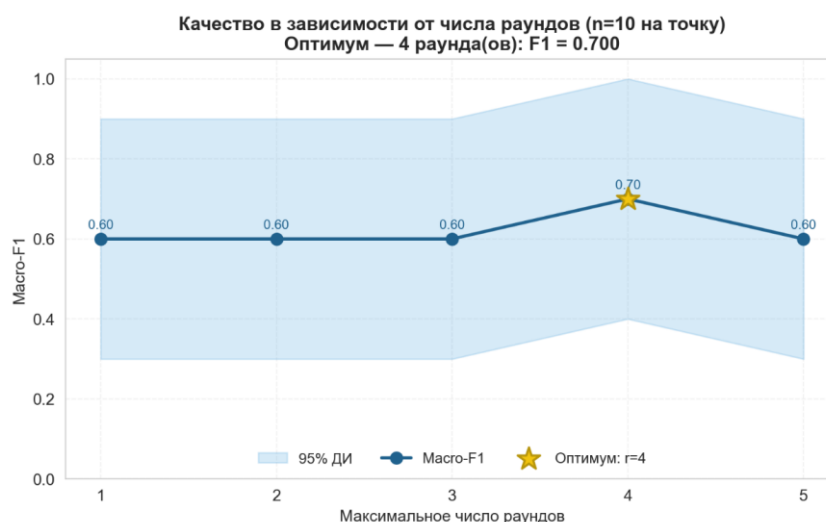


Рис. 2. Влияние числа раундов в циклической архитектуре на макро-F1 и стоимость на тест

В проведенном эксперименте значения макро-F1 для циклической архитектуры составляли 0,600 при 1–3 раундах, возрастали до 0,700 на 4 раундах и вновь снижались до 0,600 на 5 раундах; одновременно продолжали расти стоимость и время ответа. Стоимость увеличивалась от 0,0485 до 0,0847 USD на тест, а средняя длительность ответа – от 158,5 до 301,8 секунды. Следовательно, число раундов следует рассматривать как ограничиваемый параметр, а не как величину, которую нужно автоматически увеличивать. Для СППР это означает, что дополнительные итерации оправданы только там, где потенциальный прирост качества действительно перекрывает рост затрат и времени ответа.

3. Заключение

В рамках проведенного исследования установлено, что архитектура взаимодействия между агентами является существенным фактором качества решения в многоагентных дебатах крупных языковых моделей. Наибольшее наблюдаемое значение макро-F1 показала архитектура с исключением модели, однако базовая архитектура сохранила преимущество по макро-средней абсолютной ошибке и оказалась сильным компромиссным вариантом при учете стоимости и времени ответа. Следовательно, многоагентные дебаты нельзя рассматривать как гарантированный способ улучшения ответа. Их эффективность зависит от топологии

взаимодействия, типа проверяемого материала, допустимых затрат и цены ошибки. Тем самым достигнута цель исследования: определены не только различия между архитектурами, но и условия, при которых их применение в СППР является рациональным.

Практическая применимость результатов состоит в возможности выбора архитектуры под конкретный сценарий СППР. Если приоритетом является максимизация качества и допустимы дополнительные затраты, наиболее оправданной выглядит архитектура с исключением модели. Если критичны стоимость и скорость ответа, сильным рабочим решением остается базовая архитектура. В режиме активного поиска дефектов допустимы более чувствительные схемы, поскольку в этом случае важнее не пропустить реальную ошибку. В режиме осторожной проверки готового банка заданий, напротив, следует избегать архитектур с выраженной асимметрией между корректными тестами и тестами с внесенными ошибками, так как они могут создавать лишние ложноположительные решения. Следовательно, дебаты целесообразно применять в спорных и ответственных случаях: при высокой цене

пропуска ошибки, необходимости дополнительной проверки сомнительных заданий и наличии бюджета на более длительную обработку. От сложных дебатных схем следует отказываться при массовой первичной проверке, жестких ограничениях по времени ответа, низкой цене ошибки и отсутствии значимого выигрыша по качеству.

Ограничения исследования связаны с одной предметной областью, фиксированным ансамблем из трех моделей и ограниченной статистической мощностью части сравнений. Кроме того, в тех схемах, где суждения одних моделей воздействуют на итог других, нельзя игнорировать известные риски смещений, возникающих при использовании языковой модели в роли судьи [3, 4]. Поэтому полученные выводы не следует автоматически переносить на иные области, языки или составы ансамблей. Перспективными направлениями дальнейшей работы являются расширение корпуса, внешняя экспертная валидация, отдельные эксперименты с парными и более крупными ансамблями, а также проверка архитектур на других задачах поддержки принятия решений.

Comparative analysis of multi-agent debate architectures of large language models in decision support

Sh. M. Magomedov

Abstract. The paper presents a comparative analysis of five multi-agent debate architectures of large language models in a decision-support task related to evaluating the quality of educational tests. The study aims to identify how communication topology affects decision quality, computational cost, latency, and robustness. A unified experimental setting was used to compare a baseline, linear, cyclic, cross, and model-elimination architecture. The corpus comprised 60 tests from four school disciplines split into correct and errors subsets, while the model ensemble included Qwen 2.5 72B Instruct, Grok 4.1 Fast, and Gemma 4 26B-A4B IT. The main evaluation metrics were macro-F1 for detecting problematic questions, macro-MAE for the numerical assessment of test quality, and token usage, cost per test, and response time for operational comparison. The results show that the model-elimination architecture achieved the highest observed macro-F1 within the experiment, while the baseline architecture remained superior in macro-MAE and cost-efficiency. The study also demonstrates that some debate architectures are highly sensitive to the type of test and that increasing the number of rounds does not guarantee sustained quality improvement. The findings support treating debate architecture as a controllable design parameter of decision-support systems rather than as a neutral implementation detail.

Keywords: large language models, multi-agent debate, decision support, educational tests, LLM-as-a-Judge, macro-F1, interaction architecture

Литература

1. Du Y., Li S., Torralba A., Tenenbaum J. B., Mordatch I. Improving factuality and reasoning in language models through multiagent debate [Электронный ресурс] // arXiv.org. – 2023. – arXiv: 2305.14325. – URL: <https://arxiv.org/abs/2305.14325> (дата обращения: 20.02.2026). – Текст: электронный.
2. Li Y., Du Y., Zhang J. [et al.]. Improving multi-agent debate with sparse communication topology // Findings of the Association for Computational Linguistics: EMNLP 2024. – 2024. – P. 7281–7294. – URL: <https://arxiv.org/abs/2406.11776> (дата обращения: 30.06.2026).
3. Zheng L., Chiang W.-L., Sheng Y. [et al.]. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena [Электронный ресурс] // arXiv.org. – 2023. – arXiv: 2306.05685. – URL: <https://arxiv.org/abs/2306.05685> (дата обращения: 22.02.2026). – Текст: электронный.
4. Zhu L., Wang X., Wang X. JudgeLM: fine-tuned large language models are scalable judges [Электронный ресурс] // arXiv.org. – 2023. – arXiv: 2310.17631. – URL: <https://arxiv.org/abs/2310.17631> (дата обращения: 25.02.2026). – Текст: электронный.
5. Guo T., Chen X., Wang Y. [et al.]. Large language model based multi-agents: a survey of progress and challenges // Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence. – 2024. – URL: <https://arxiv.org/pdf/2402.01680> (дата обращения: 30.06.2026).
6. Shi Y., Liang R., Xu Y. EducationQ: evaluating LLMs' teaching capabilities through multi-agent dialogue framework // Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics. – 2025. – P. 32799–32828. – URL: <https://arxiv.org/html/2504.14928> (дата обращения: 30.06.2026).
7. Аванесов В. С. Форма тестовых заданий: учебное пособие для учителей школ, лицеев, преподавателей вузов и колледжей. – 2-е изд., перераб. и расшир. – Москва: Центр тестирования, 2005. – 156 с.
8. Чельшкова М. Б. Теория и практика конструирования педагогических тестов. – Москва: Логос, 2002. – 432 с.
9. Приказ Министерства просвещения Российской Федерации от 31.05.2021 № 287 «Об утверждении федерального государственного образовательного стандарта основного общего образования» [Электронный ресурс]. URL: <https://publication.pravo.gov.ru/Document/View/0001202107050027> (дата обращения: 13.03.2026). – Текст: электронный.
10. Федеральный институт педагогических измерений. Аналитические и методические материалы по разработке и оцениванию контрольных измерительных материалов [Электронный ресурс]. – URL: <https://fipi.ru/ege/analiticheskie-i-metodicheskie-materialy> (дата обращения: 13.03.2026). – Текст: электронный.

Поступила в редакцию / Received: 15.06.2026.

Поступила после рецензирования / Revised: 05.06.2026.

Принята к печати / Accepted: 06.07.2026.

УДК 621.3.049.75

Обеспечение устойчивого электропитания печатной платы микропроцессорной системы при подключении через длинный кабель

И. А. Лафазан¹¹НИЦ «Курчатовский институт» – НИИСИ, Москва, lafazan@cs.niisi.ras.ru

Аннотация. В статье рассматривается задача формирования устойчивого входного узла питания печатной платы микропроцессорной системы, подключённой к источнику питания через длинный кабель. Показано, что в этих условиях применение дорогостоящих керамических конденсаторов большой ёмкости без учёта демпфирования может приводить к неустойчивому переходному процессу, поскольку индуктивность кабеля совместно со входной ёмкостью образуют резонансный контур достаточно высокой добротности. Предложен подход к построению входного узла питания, обеспечивающий демпфирование резонансных процессов без чрезмерной просадки напряжения и перегрева, шунтирование высокочастотных составляющих тока и локализацию помех. Основная идея подхода – использовать ESR накопительного входного конденсатора как средство демпфирования резонанса, а влияние последовательной индуктивности компенсировать керамическими конденсаторами малых номиналов. Показано, что аналогичный принцип неприемлем в локальных цепях развязки микросхем. Приведены рекомендации по выбору диапазона значений эквивалентного последовательного сопротивления, номиналов входных конденсаторов и размещению элементов на печатной плате.

Ключевые слова: печатная плата, электропитание, добротность, входной фильтр, резонанс

1. Введение

Современные микропроцессорные системы (МПС) характеризуются импульсным потреблением тока, поэтому система подачи питания должна рассматриваться как часть электрической схемы устройства. Качество электропитания печатной платы МПС определяется не только параметрами источника, но и паразитными сопротивлениями и индуктивностями соединителей, кабелей, печатной платы и пассивных компонентов. В известных рекомендациях по построению цепей питания основное внимание обычно уделяется снижению импеданса питания, локальной развязке микросхем, применению низкоимпедансных многослойных керамических конденсаторов (multi-layer ceramic capacitor, MLCC) и подавлению высокочастотных помех ферритовыми бусинами. При этом вопросы формирования устойчивого входного узла питания печатной платы при подключении к удалённому источнику через протяжённую кабельную линию обычно рассматриваются значительно реже.

Традиционный подход, как правило, предполагает установку на входе платы

низкоимпедансных керамических конденсаторов, а при необходимости – последовательного ферритового элемента или дросселя, при обязательной локальной развязке микросхем. Он эффективен для уменьшения высокочастотного импеданса и подавления части наводимых помех, однако не является универсальным решением при питании платы от удалённого источника через длинный кабель.

Практическая значимость задачи определяется широким применением распределённых электронных систем, в которых источник питания удалён от печатной платы и соединён с ней кабелем или жгутом значительной длины. Такая ситуация характерна для промышленной автоматики, измерительных комплексов, телеметрии, робототехники, транспортной и бортовой аппаратуры, модулей связи, систем управления и удалённых датчиковых узлов. В составе перечисленного оборудования МПС чувствительны даже к кратковременным провалам, выбросам и колебаниям напряжения питания, которые могут вызывать сброс, нарушение обмена по интерфейсам, ошибки измерений и ложное срабатывание защит. Подобная конфигурация возникает не только в штатной эксплуатации

распределённых систем, но и при стендовых и климатических испытаниях, когда источник питания выносится за пределы печатной платы.

При подаче питания через длинный кабель его индуктивность становится существенной частью входного контура и совместно со входной ёмкостью платы образует резонансную систему. В результате при включении и скачкообразных изменениях нагрузки на входе платы могут возникать провалы, выбросы и автоколебательный переходный процесс («звон»). При этом критичным становится не только обеспечение требуемого уровня постоянного напряжения, но и форма переходного процесса на входе устройства. В этих условиях простое наращивание дорогостоящей низкоимпедансной керамической ёмкости не всегда улучшает поведение системы и может повышать добротность контура, а ферритовые элементы, эффективные преимущественно в высокочастотной области, не обеспечивают должного уровня демпфирования низкочастотного резонанса. Дополнительно выбор решения ограничивается высокой токовой нагрузкой, компоновкой, доступностью компонентов и бюджетом.

В связи с этим важной задачей работы является подбор конструкции и параметров входного узла питания печатной платы, подключённой к удалённому источнику через длинный кабель, при котором обеспечиваются устойчивый переходный процесс и снижение высокочастотных токов в кабельной линии. Цель статьи – проанализировать особенности системы «длинный кабель – входная ёмкость», показать ограничения традиционных входных решений и сформулировать практические рекомендации по построению входного узла питания печатной платы.

2. Ограничения типовых решений входной фильтрации при питании через длинный кабель

Как отмечалось ранее, при питании печатной платы МПС от удалённого источника через длинный кабель типовые решения входной фильтрации, ориентированные главным образом на снижение высокочастотного импеданса и подавление проводимых кондуктивных помех, не всегда обеспечивают требуемое поведение входного узла. На практике широко применяется схема, основанная на включении в разрыв линии последовательного ферритового элемента или

дросселя и установке на входе платы низкоимпедансных керамических конденсаторов. Однако при подключении через длинную кабельную линию эффективность работы такой системы ограничивается рядом физических причин и проектных факторов, таких как:

- высокие токи нагрузки, протекающие по кабельной линии электропитания и входным цепям печатной платы ограничивают применение компонентов, включаемых в разрыв линии из-за дополнительного падения напряжения и тепловыделения;
- снижение высокочастотного импеданса ферритовых элементов при протекании постоянного тока нагрузки: смещение рабочей точки материала по магнитной характеристике уменьшает эффективную магнитную проницаемость и ухудшает фильтрующие свойства элемента;
- невозможность применения выводных и крупногабаритных компонентов для монтажа в отверстия печатной платы (through-hole technology, THT) из-за ограничений компоновки;
- необходимость точного подбора параметров LC/RLC-фильтра и последующей экспериментальной верификации с учётом паразитных параметров кабеля, монтажа и компонентов; растущие требования по помехоустойчивости и электромагнитной совместимости;
- снижение активного сопротивления кабельной линии уменьшает потери в контуре «кабель – входная ёмкость» и тем самым повышает его добротность;
- малые значения эквивалентного последовательного сопротивления (equivalent series resistance, ESR) современных MLCC, снижающие потери в контуре «кабель – входная ёмкость» и тем самым повышающие его добротность;
- малая эквивалентная последовательная индуктивность таких конденсаторов дополнительно расширяет область низкого импеданса, но не обеспечивает демпфирование резонанса;
- потеря фактической ёмкости многослойного керамического конденсатора при приложении постоянного напряжения (DC-bias)[1];
- возможность возникновения антирезонансных пиков импеданса при параллельном включении керамических конденсаторов разных номиналов и типоразмеров вследствие различия их собственных резонансных частот;

- конструктивные ограничения, связанные с плотной компоновкой, механической надёжностью, минимизацией числа компонентов и снижением паразитной индуктивности соединений;
- интенсивное импортозамещение электронных модулей и ограниченная номенклатура ферритовых бусин у отечественных производителей;
- ограниченность бюджета.

Таким образом, типовой подход к формированию входного узла питания по схеме «последовательная ферритовая бусина или дроссель + MLCC на общий провод» не может рассматриваться как универсальное решение для модуля, питаемого через длинный кабель. Импеданс ферритовых бусин, как правило, нормируется на частоте 100 МГц, а область их эффективного применения обычно соответствует диапазону 10-300 МГц. В то же время частота основного резонанса контура «кабель – входная ёмкость» при типичных значениях индуктивности кабельной линии 10-50 мкГн и входной ёмкости 10-470 мкФ находится в диапазоне 1-17 кГц. Поэтому низкочастотный звон и выбросы, обусловленные высокой добротностью контура, могут сохраняться даже независимо от наличия ферритовой бусины.

Более того, добавление только MLCC без средств демпфирования уменьшает импеданс входного узла [2], но из-за малого эквивалентного последовательного сопротивления увеличивает добротность колебательной системы. Следовательно, при построении входного узла питания необходимо учитывать не только требования к высокочастотному шунтированию вблизи разъёма питания, но и необходимость введения контролируемых потерь в низкочастотный резонансный контур.

В работе предлагается использовать ESR накопительного входного конденсатора как средство демпфирования резонанса, а влияние последовательной индуктивности компенсировать керамическими конденсаторами малых номиналов. Такой подход позволяет преодолеть или заметно ослабить перечисленные ограничения, упростить закупку и подбор компонентов, повысить технологичность электронного модуля и сделать его выпуск более

экономически обоснованным.

3. Создание устойчивого входного узла МПС

3.1 Расчётная модель контура «длинный кабель – входная ёмкость»: просадка, тепловыделение и резонанс

Разберём подробнее процессы, которые будут возникать в системе «длинный кабель – входная ёмкость», схематично изображённой на рисунке 1.

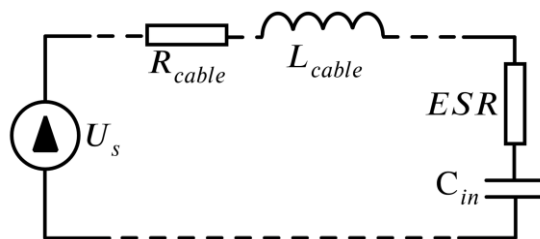


Рис.1. Схематичное изображение системы «длинный кабель – входная ёмкость»

Для первичного анализа кабельную линию удобно представить сосредоточенными элементами: сопротивлением R_{cable} , и индуктивностью L_{cable} на входе платы стоит накопительный конденсатор, ёмкостью C_{in} , а питание осуществляется источником напряжения U_s . В данной модели последовательные индуктивности входного конденсатора и печатных проводников не учитываются, поскольку для рассматриваемого низкочастотного резонанса их величины существенно меньше индуктивности L_{cable} протяжённой кабельной линии [3].

$$U_s = U_c(t) + I(t)(R_{cable} + ESR) + L_{cable} * \frac{dI(t)}{dt}, \quad (1)$$

Где $I(t) = C_{in} * \frac{dU_c(t)}{dt}$ – ток зарядки печатной платы, а $U_c(t)$ – напряжение непосредственно на входной ёмкости C_{in} .

В момент подачи питания входная ёмкость эквивалентна низкоомной нагрузке, а ток её зарядки растёт со скоростью, ограниченной L_{cable} . Типичная форма переходного процесса подачи питания при длине кабеля 15 м и напряжении 48 В представлена на рисунке 2.



Рис. 2. Изменение напряжения на входе при подаче питания напряжением 48 В через кабель длиной 15 м [4], отражающее неустойчивость системы

Для оценки влияния ESR необходимо рассматривать не только общий рост напряжения на входе, но и мгновенное распределение напряжения в последовательном контуре «кабель – входная ёмкость». При протекании тока заряда входной ёмкости на эквивалентном последовательном сопротивлении входного конденсатора возникает падение напряжения:

$$\Delta U_{ESR} = I(t) * ESR, \quad (2)$$

Соответственно, пусть ΔU_{MAX} – допустимая величина максимальной просадки напряжения, то при пиковом токе I_{MAX} :

$$ESR \leq \frac{\Delta U_{MAX}}{I_{MAX}}, \quad (3)$$

Если этим правилом пренебречь и поставить конденсатор со слишком большим ESR , это приведёт к избыточной просадке напряжения.

Также мгновенная мощность, выделяющаяся при подаче на конденсатор напряжения (зарядке конденсатора) и резких скачках тока потребления:

$$P(t) = I^2 * ESR, \quad (4)$$

В результате чего на конденсаторе теряется часть мощности. Если импульсы будут частыми, конденсатор ощутимо нагреется при протекании скачков тока через ESR .

Далее, по завершении первичного процесса заряда входных ёмкостей система «кабель – входная ёмкость» ведёт себя как колебательный

контур, с резонансной частотой:

$$f_{rez} = \frac{1}{2\pi\sqrt{L_{cable}C_{in}}}, \quad (5)$$

И добротностью:

$$Q = \frac{1}{R_{cable} + ESR + R_s} \sqrt{\frac{L_{cable}}{C_{in}}}, \quad (6)$$

Где R_s – выходное сопротивление источника питания.

3.2 Последствия высокой добротности и выбор диапазона ESR накопительных ёмкостей

При малом суммарном последовательном сопротивлении контура «кабель – входная ёмкость» его добротность возрастает, а энергия, запасённая в L_{cable} и C_{in} , рассеивается медленно. В результате переходный процесс приобретает выраженный колебательный характер. Основные последствия такого режима следующие:

- 1) Выбросы и провалы напряжения в цепях питания печатной платы. (Рис.3).

При этом ток в кабельной линии может иметь колебательный характер и на отдельных интервалах менять направление, что соответствует возврату части энергии, накопленной во входной ёмкости, в сторону источника питания [5]. Поэтому переходный процесс сопровождается не только провалом

при заряде ёмкости, но и последующими выбросами напряжения на входе устройства. Примеры таких переходных процессов приведены на рисунке 3.

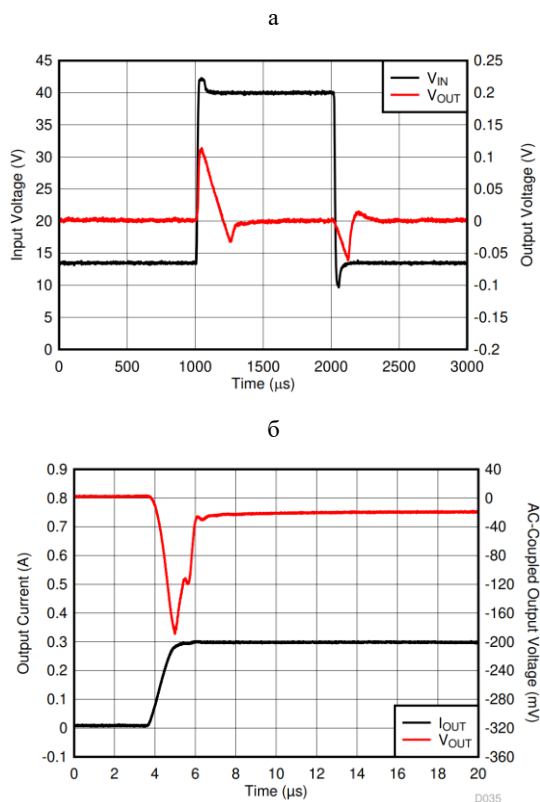


Рис.3. Примеры переходных процессов в цепях питания:

- а) реакция выходного напряжения стабилизатора на резкое изменение входного напряжения. Чёрная кривая соответствует входному напряжению, а красная — отклонению выходного напряжения от установившегося значения [6];
- б) провал выходного напряжения при скачкообразном увеличении тока нагрузки [7]

Провалы могут приводить к пересечению порога блокировки при пониженном напряжении питания (UVLO, undervoltage lockout), а также порога срабатывания цепи сброса, что вызывает ложные перезапуски цифровых устройств. Выбросы – к превышению номинального напряжения микросхем и входных преобразователей.

- 2) Увеличение длительности переходных процессов.

Поскольку энергия колебаний накапливается в L_{cable} и C_{in} при малых потерях контур затягивает установление напряжения (длительность переходного процесса может составлять от нескольких периодов собственных колебаний до значительно большего времени при взаимодействии с цепями запуска, UVLO,

ограничением тока и алгоритмами защиты источника питания), что ухудшает повторяемость запуска и может приводить к циклическим перезапускам источника питания или преобразователей на плате.

- 3) Рост циркулирующих токов и локальный перегрев.

При звоне возникают переменные токи, протекающие между кабелем и входной ёмкостью. Даже если средний ток нагрузки невелик, действующее значение переменной составляющей может быть значительным, что увеличивает потери на R_{cable} , ESR , контактных сопротивлениях разъёмов и повышает тепловую нагрузку элементов.

- 4) Ухудшение электромагнитной совместимости.

Колебательный ток в кабеле увеличивает проводимые помехи и может повышать излучение за счёт превращения кабеля в эффективную антенну, особенно при наличии более высокочастотных составляющих, возникающих из-за нелинейностей нагрузки и коммутационных процессов.

Если источник питания расположен далеко от устройства, длина, а, следовательно, и индуктивность L_{cable} неизбежно будут большими.

Теоретически, ёмкость входных конденсаторов C_{in} можно увеличить, но это даст:

- большой скачок тока коммутации, который не все компоненты схемы могут выдержать;
- сделает запуск более жёстким, долгим и нестабильным, сработает защита и будет перезапуск;
- по формуле (5) уменьшение резонансной частоты, переходные процессы станут более долгими;
- усложнение поиска подходящего под большую ёмкостную нагрузку источника питания;
- увеличение энергии колебаний $E_c = 0.5CU^2$, а, следовательно, и циркулирующих токов;
- увеличение габаритных размеров конденсатора;
- увеличение закупочной стоимости компонента.

Ещё одним способом уменьшения резонанса и снижения амплитуды колебаний является увеличение сопротивления силового кабеля R_{cable} или выходного сопротивления источника питания R_s . Но так как эти сопротивления работают одинаково на всех частотах, в случае их увеличения произойдёт значительная потеря энергии на нагрузке, а также дополнительное падение напряжения.

Главным параметром, который предложено подобрать, чтобы уменьшить добротность колебательного контура, является ESR входного конденсатора. Для достижения оптимально высокого коэффициента затухания требуется обеспечить соотношение $Q \leq 0,5$ [8] (если взять меньше, то система будет передемпфированной [9]). Исходя из этого получаем:

$$ESR \geq 2 \sqrt{\frac{L_{\text{cable}}}{C_{\text{in}}}} - R_{\text{cable}} - R_s, \quad (7)$$

Из формулы (7) видно, что при отрицательной правой части условие будет выполняться, даже если ESR входной ёмкости будет равен нулю, поэтому при невозможности уменьшения активного сопротивления нет необходимости специально вводить ESR для затухания резонанса и следует ориентироваться на минимально возможное ESR , чтобы не увеличить просадку напряжения и нагрев.

Преимуществом использования ESR накопительного конденсатора в качестве демпфирующего параметра является то, что потери в конденсаторе проявляются преимущественно в переменной составляющей тока, не увеличивая сопротивление линии по постоянному току. При этом ESR является частотно-зависимой величиной и в технической документации часто нормируется на частоте 100 кГц (иногда также приводятся графики $ESR(f)$). Для большинства практических конфигураций «длинный кабель – входная ёмкость» резонансная частота лежит в области, где паспортные данные ESR позволяют выполнить первичную инженерную оценку демпфирования и оценить добротность. Окончательная оценка выполняется по осциллограммам/модели с учётом $ESR(f)$ (при его наличии в технической документации), ESL и параметров кабеля.

Таким образом, ESR входной накопительной ёмкости должен находиться в допустимом диапазоне: нижняя граница определяется требованием демпфирования резонанса контура «кабель – входная ёмкость» (7), а верхняя – допустимой просадкой напряжения и тепловой нагрузкой на конденсатор (3), (4). Увеличение ESR действительно уменьшает пиковый ток заряда входной ёмкости и снижает добротность контура, однако чрезмерно большое ESR приводит к увеличению падения напряжения при скачках тока, росту потерь и ухудшению энергетической поддержки нагрузки. Поэтому ESR не следует рассматривать как универсальное средство ограничения пускового тока. Если требуемая входная ёмкость вызывает недопустимо большой ток включения, целесообразно применять активное ограничение

пускового тока: схемы плавного пуска, контроллеры горячего подключения, электронные предохранители или МОП-транзисторные ключи с управляемым нарастанием тока. Такие решения позволяют ограничить пусковой ток без постоянного увеличения сопротивления силовой линии и без ухудшения динамических свойств входного узла в рабочем режиме.

Практика задания допустимого диапазона ESR не нова и широко известна по линейным стабилизаторам. В ряде линейных стабилизаторов напряжения ESR выходного конденсатора является частью частотной коррекции: совместно с выходной ёмкостью он формирует частотную коррекцию, необходимую для устойчивой работы петли обратной связи [10]. Поэтому техническая документация нередко ограничивает ESR снизу и сверху; слишком низкий ESR (например, при замене танталового конденсатора на более технологичный и дорогой MLCC без учёта компенсации, что на первый взгляд, кажется, улучшит фильтрацию) может вызвать колебательный режим или самовозбуждение, что проявляется ростом пульсаций выходного напряжения. Пример графика зависимости ESR от выходного тока с указанным оптимальным диапазоном представлен на рисунке 4.

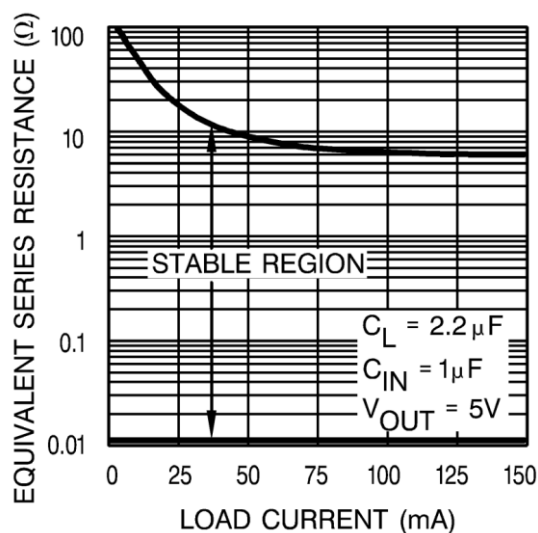


Рис.4. Допустимый диапазон ESR выходного конденсатора ЛДО регулятора LP2950/LP2951 [11]

Сходный инженерный принцип используется и при подборе входных накопительных ёмкостей: ESR входного накопительного конденсатора рассматривается не как нежелательный паразитный параметр, который следует минимизировать, а как элемент, целенаправленно формирующий динамическое поведение системы питания. Как

и в случае выходных конденсаторов линейных стабилизаторов, *ESR* участвует в частотной характеристике узла питания и влияет на склонность системы к колебательным процессам. Поэтому его значение должно выбираться не по принципу минимизации, а исходя из требуемого демпфирования.

3.3 Область применения и ограничения ферритовой бусины

Ещё одним методом фильтрации входной цепи питания является установка ферритовой бусины последовательно в цепь питания электронного модуля совместно с керамическими конденсаторами с низким *ESR*. Конструктивно бусина – это ферритовый сердечник (материал с магнитными потерями), через который проходит проводник (в компонентах поверхностного монтажа это ферритовый корпус с внутренним проводящим каналом). Действует она по сути как частотно-зависимый импеданс с существенной активной составляющей, за счёт которой энергия высокочастотных помех рассеивается в тепло. На низких частотах её импеданс мал и определяется в основном сопротивлением проводящего канала и небольшой индуктивной составляющей. В рабочем частотном диапазоне возрастает вклад магнитных потерь в феррите, поэтому активная составляющая импеданса становится существенной и обеспечивает подавление высокочастотных помех. На частотах выше рабочего диапазона влияние паразитной ёмкости приводит к снижению импеданса бусины и ухудшению её подавляющих свойств.

В технической документации ферритовых бусин даны графики зависимости модуля импеданса, реактивной и активной части сопротивления от частоты. Получается, что данный компонент эффективен в случае, если шум высокочастотный (порядка 100 МГц) и задача его рассеять в тепло, а не отфильтровать идеальной индуктивностью (дресселем), которая будет только накапливать и отдавать энергию, порождая резонанс. Однако при подключении по длинному кабелю доминирующим паразитным параметром становится индуктивность линии, и основной переходный процесс определяется резонансом контура «кабель – входная ёмкость», частота которого лежит в килогерцовом диапазоне. А значит, ферритовая бусина не обеспечит должного уровня демпфирования и не поможет решить проблему звона и выбросов, обусловленных большой добротностью паразитного колебательного контура. Более того, зачастую её применение несёт в себе ряд недостатков:

1) Ограничение по току и нагрев элемента. Допустимый DC-ток ограничен потерями и допустимым перегревом корпуса. Для компактных SMD-типоразмеров ток часто составляет единицы ампер (типично 1–6 А; значения 10–15 А встречаются, но это уже крупные/дорогие позиции). При высоких токах возрастает падение напряжения и тепловая нагрузка на компонент.

2) Снижение импеданса при постоянной составляющей тока (DC bias).

Постоянный ток смещает феррит по магнитной характеристике, и высокочастотный импеданс (в первую очередь реактивная составляющая) уменьшается. В результате фильтрующие свойства бусины на практике могут оказаться хуже, чем ожидалось. Это особенно критично для высоконагруженных входных цепей.

3) Потенциальное образование дополнительных резонансов.

В сочетании с низкоимпедансными керамическими конденсаторами бусина может формировать новый резонанс/антирезонанс (из-за своего частотно-зависимого $Z(f)$, паразитной ёмкости и паразитной индуктивности конденсаторов), что усложняет обеспечение устойчивого поведения входного узла.

4) Стоимость и доступность силовых бусин

Ферритовые бусины, рассчитанные на большие токи и обладающие требуемыми частотными характеристиками, могут иметь высокую стоимость и ограниченную доступность. В ряде случаев подбор отечественных аналогов с требуемыми параметрами затруднён.

На данный момент закупка отечественных ферритовых бусин может быть затруднительна.

При этом ферритовые бусины остаются эффективным средством подавления проводимых высокочастотных помех небольших токов и могут применяться как дополнительная мера при стремлении выполнения требований по ЭМС. Однако, в любом случае, обеспечение аperiodического переходного процесса при длинном кабеле требует отдельного демпфирования низкочастотного контура. С этим как раз могут справиться конденсаторы, с *ESR*, находящимся в оптимальном диапазоне.

3.4 Высокочастотное шунтирование и локальная развязка микросхем

Но у накопительного конденсатора, помимо паразитного сопротивления *ESR*, есть ещё паразитная индуктивность. Она в контексте фильтрации входного питания никак не

помогает, а только препятствует замыканию токов высокой частоты (быстрых фронтов) локально рядом с разъёмом на землю. Если в цепи не стоит ферритовой бусины и отсутствует в/ч-шунтирование у разъёма, эти токи будут создавать шумы и помехи на печатной плате, а также выходить за её пределы по кабелю, ухудшая ЭМС. Поэтому параллельно накопительным конденсаторам с относительно большими паразитными параметрами обязательна установка одного или нескольких керамических MLCC малой ёмкости (порядка 10 нФ-1 мкФ), ESR которых минимальны.

Эффективная область работы каждого MLCC определяется его собственной резонансной частотой (serial resonance frequency, SRF) и индуктивностью монтажной петли. На частотах выше SRF конденсатор становится индуктивным, поэтому критично минимизировать индуктивность подключения (короткие соединения, близкорасположенные переходные отверстия, малая площадь петли тока, установка максимально близко к разъёму). Для типовых корпусов 0402 – 0805 стандарта EIA 0402-0805 (1005M-2012M ГОСТ/МЭК) конденсаторы порядка 10-100 нФ обеспечивают низкий импеданс в диапазоне десятков-сотен МГц, а 0.47-1 мкФ – в области единиц – десятков МГц. Применение нескольких

параллельно подключённых MLCC разных номиналов позволяет расширить диапазон низкого импеданса и улучшить локализацию высокочастотных токов у разъёма, но одновременно увеличивает риск возникновения антирезонансных пиков. Поэтому количество номиналов и их типоразмеры следует выбирать с учётом частотного состава помех, параметров монтажа и результатов моделирования или измерений.

Необходимо отметить, что входной фильтр на разъёме питания не является заменой локальным блокировочным шунтирующим по переменному току (bypass capacitors) конденсаторам, поскольку эти элементы решают разные задачи. Входной фильтр формирует требуемое поведение питания на границе устройства: демпфирует резонансы, снижает проводимые помехи и ограничивает проникновение высокочастотных токов в кабель и их выход за пределы МПС. Локальные блокировочные конденсаторы, напротив, обеспечивают замыкание импульсных токов потребления непосредственно вблизи микросхем. Они уменьшают высокочастотные просадки питания в точке нагрузки, сокращают площадь токовой петли и тем самым снижают наводки и излучение, как показано на рисунке 5.

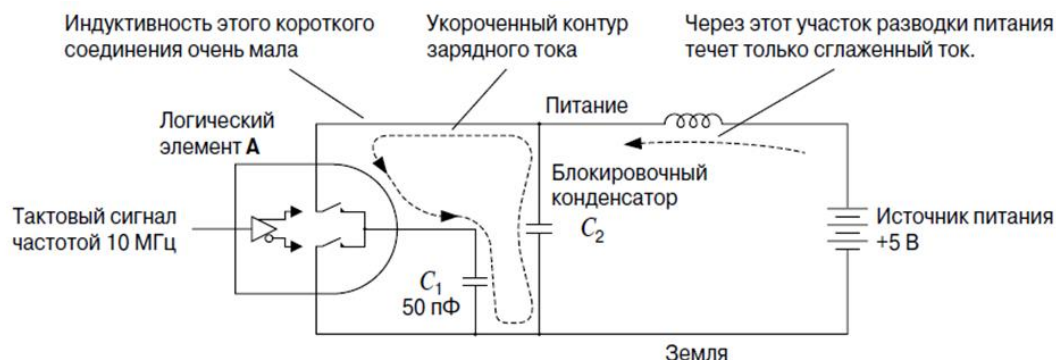


Рис. 5. Сокращение петли импульсного тока за счёт локального блокировочного конденсатора C_1

В этой задаче любые паразитные параметры (прежде всего ESL и ESR самого конденсатора, а также индуктивность и сопротивление дорожек/переходных отверстий) работают только во вред, поскольку приводят к дополнительным выбросам на быстрых фронтах и мгновенным просадкам, а также снижают эффективный диапазон частот байпаса (выше собственной резонансной частоты конденсатор становится индуктивным). Поэтому bypass-конденсаторы устанавливают максимально близко к выводам питания микросхем (обычно по одному на каждый вывод питания или на группу выводов), с минимальной

индуктивностью подключения (короткие соединения, прямой доступ к плоскостям питания/земли, отдельные переходные рядом с площадками). Для блокировочных конденсаторов некорректно задавать целевой диапазон ESR , как это было предложено для демпфирования входного контура. Наоборот, требуется минимизировать суммарный импеданс пути в рабочем диапазоне частот. Выбор номиналов и типоразмера байпасов определяется спектром токовых импульсов (крутизной фронтов и рабочей частотой), допустимыми пульсациями на шине и достижимой индуктивностью монтажа.

На практике применяют минимально достаточный набор MLCC разных номиналов, расширяющий полосу низкого импеданса без избыточного усложнения и риска выраженных антирезонансов.

Также не следует думать, что высокотехнологичные керамические конденсаторы больших ёмкостей не имеют смысла и всегда могут быть заменены на другие, с более высокими паразитными параметрами. Низкие *ESR* и паразитная индуктивность MLCC обеспечивают минимальный импеданс на средних и высоких частотах, благодаря чему они широко применяются для подавления быстрых просадок питания и локализации высокочастотных токов вблизи потребителя импульсного тока, источника помех или входного разъёма – в зависимости от решаемой задачи. В системах без длинных кабелей, а также при корректно реализованном демпфировании входного контура использование больших MLCC может улучшить динамику питания и снизить уровень шумов с учётом снижения эффективной ёмкости MLCC под действием постоянного смещения.

3.5 Предложенная структура входного узла

Предлагаемая структура входного узла основана на функциональном разделении элементов по частотным диапазонам и на целенаправленном использовании паразитных параметров компонентов, которые в типовом подходе обычно стремятся минимизировать. Непосредственно у входного разъёма платы устанавливается накопительный конденсатор, эквивалентное последовательное сопротивление которого выбирается не по принципу минимизации, а в расчётном диапазоне, обеспечивающем демпфирование низкочастотного резонанса контура «кабель – входная ёмкость» при допустимых просадке напряжения и тепловыделении. За счёт этого накопительный конденсатор выполняет одновременно две функции: служит локальным энергетическим резервом и вносит контролируемые потери в резонансный контур, переводя переходный процесс в апериодический или слабоколебательный режим без введения дополнительного последовательного сопротивления в силовую линию.

Параллельно накопительному конденсатору у входного разъёма размещаются керамические конденсаторы MLCC малой и средней ёмкости с минимальной индуктивностью подключения. Их назначение состоит не в демпфировании низкочастотного резонанса, а в шунтировании высокочастотных составляющих тока, локализации токовых петель вблизи входа

платы и снижении проникновения помех в кабельную линию. Далее по плате локальные блокировочные конденсаторы у микросхем замыкают импульсные токи непосредственно в точках потребления. Тем самым входной узел получает согласованную структуру, в которой накопительные конденсаторы формируют устойчивое поведение системы в области низких частот, а MLCC обеспечивают низкий импеданс в средней и высокочастотной области.

Принципиально важным является то, что устойчивость входного узла достигается не простым наращиванием низкоимпедансной керамической ёмкости и не обязательным включением последовательной ферритовой бусины или дросселя, а согласованием параметров кабеля, источника питания, накопительной ёмкости и нагрузки. Такой подход позволяет во многих практических случаях снизить зависимость решения от дефицитных токо-ограниченных последовательных компонентов, упростить компоновку и получить предсказуемую форму переходного процесса при сохранении требуемого уровня высокочастотного шунтирования.

4. Заключение

В статье рассмотрена задача формирования устойчивого входного узла высоконагруженного питания печатной платы МПС при подключении через длинный кабель в условиях ограниченной доступности компонентной базы. Показано, что индуктивность кабеля и входная ёмкость образуют резонансный контур, который, при недостаточном демпфировании, вызывает звон, провалы/выбросы напряжения, рост циркулирующих токов и ухудшение ЭМС. Установлено, что простое наращивание количества низкоимпедансных керамических ёмкостных элементов может повышать добротность контура и усиливать колебания, а применение ферритовых бусин не всегда эффективно из-за частотного несоответствия, токовых ограничений и влияния DC-bias.

Предложенное решение основано на разделении функций компонентов: bulk-конденсатор подбирается таким образом, чтобы его *ESR* попадал в диапазон, обеспечивающий демпфирование низкочастотного резонанса при допустимой просадке напряжения, а MLCC малой/средней ёмкости у разъёма обеспечивают локализацию высокочастотных токов и снижение импеданса в частотной области, где эффективность bulk-конденсатора ограничена его паразитной индуктивностью. Такой подход

позволяет получить апериодический или слабоколебательный переходный процесс, снизить уровень высокочастотных токов в кабеле и повысить устойчивость работы устройства при меньшей зависимости от дефицитных последовательных ферритовых элементов. Результат должен подтверждаться измерениями переходных процессов и, при необходимости, уточнением модели с учётом

$ESR(f)$, паразитной индуктивности и параметров кабеля.

Работа выполнена в рамках государственного задания НИЦ «Курчатовский институт» – НИИСИ по теме FNEF-2024-0003 этап 2026 года.

Providing stable power supply to the printed circuit board of a microprocessor system when connected via a long cable

I. A. Lafazan

Abstract. This article considers the problem of creating a stable input power supply node for a printed circuit board (PCB) of a microprocessor system connected to a power source via a long cable. It is shown that under these conditions, the use of expensive, large-capacity ceramic capacitors without taking damping into account can lead to an unstable transient process, since the cable inductance, together with the input capacitance, form a resonant circuit with a sufficiently high Q factor. An approach to constructing an input power supply node is proposed that ensures damping of resonant processes without excessive voltage drop and overheating, shunting of high-frequency current components, and localization of interference. The main idea of this approach is to use the ESR of the storage input capacitor as a means of resonance damping, and to compensate for the effect of series inductance with small-value ceramic capacitors. It is shown that a similar principle is unacceptable in local decoupling circuits of microcircuits. Recommendations are provided for selecting the range of equivalent series resistance values, the nominal values of input capacitors, and the placement of components on the PCB.

Keywords: printed circuit board, power supply, quality factor, input filter, resonance

Литература

1. Texas Instruments. LMZ21701 1-A Nano Module With 17-V Maximum Input Voltage: Datasheet [Electronic resource] // Texas Instruments : website. — 2018. — URL: <https://www.ti.com/lit/ds/symlink/lmz21701.pdf> (дата обращения: 14.05.2026).
2. Klein S. Input Filter for DCDC Converter. Design and Application Considerations of Input Filter to Reduce Conducted Emissions Caused by DC/DC Converter: Application Note No. 62AN101E Rev.003 [Electronic resource] // ROHM Co., Ltd. : website. — 2020. — URL: https://fscdn.rohm.com/en/products/databook/applinote/ic/power/switching_regulator/input_filter_for_dcdc_converter_appli-e.pdf (дата обращения: 14.05.2026).
3. H. Johnson, M. Graham. High-Speed Digital Design: A Handbook of Black Magic. Prentice Hall PTR, 1993.
4. Робертс С., Гончаров И. Решение проблем с пусковым током источников питания // Электронные компоненты. — 2020. — № 12. — С. 40–42. — URL: <https://www.elcomdesign.ru> (дата обращения: 19.03.2026).
5. Toshiba Electronic Devices & Storage Corporation. Load Transient Response of LDO and Methods to Improve It: Application Note [Electronic resource] // Toshiba : website. — 2021. — URL: https://toshiba.semicon-storage.com/info/application_note_en_20210326_AKX00312.pdf?did=66268 (дата обращения: 14.05.2026).
6. Texas Instruments. TPS7B84-Q1 150-mA, 40-V, Adjustable, Low-Dropout Regulator [Electronic resource] // Texas Instruments : website. — 2025. — URL: <https://www.ti.com/lit/gpn/tps7b84-q1> (дата обращения: 14.05.2026).

7. Texas Instruments. TPS7A20 300-mA, Ultra-Low-Noise, Low-IQ, High PSRR LDO [Electronic resource] // Texas Instruments : website. — 2024. — URL: <https://www.ti.com/product/TPS7A20> (дата обращения: 14.05.2026).
8. Baker B. Is Your Op Amp Filter Ringing? Look at Q! [Electronic resource] // Texas Instruments : website. — 2015. — URL: <https://www.ti.com/document-viewer/lit/html/ssztcml> (дата обращения: 14.05.2026).
9. «Затухающий гармонический осциллятор [Электронный ресурс] // StudFiles : сайт. — URL: <https://studfile.net/preview/15835505/page:7/> (дата обращения: 14.05.2026).
10. National Semiconductor. LM1117/LM1117I 800mA Low-Dropout Linear Regulator [Электронный ресурс] URL: <https://www.ti.com/lit/ds/symlink/lm1117.pdf> (дата обращения: 19.12.2025)
11. National Semiconductor. LP2950/LP2951 Series of Adjustable Micropower Voltage Regulators [Электронный ресурс] URL: <https://www.ti.com/lit/ds/symlink/lp2950-n.pdf> (дата обращения: 19.12.2025)

Поступила в редакцию / Received: 19.05.2026.

Поступила после рецензирования / Revised: 01.07.2026.

Принята к печати / Accepted: 01.07.2026.

УДК 004.312

Блок приема данных от внешнего АЦП с поддержкой ПДП по интерфейсу SPI

Е. А. Соловьева¹, А. В. Корниленко²

¹НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, esoloveva@cs.niisi.ras.ru

²НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, akorn@cs.niisi.ras.ru

Аннотация. Разработан сложнофункциональный блок (СФ-блок), обеспечивающий прием потоковых данных от внешнего аналогово-цифрового преобразователя (АЦП) по последовательному периферийному интерфейсу (SPI). Особенностью данного блока является то, что АЦП выступает в роли ведущего, генерируя тактовый сигнал и управляя обменом по шине SPI. Взаимодействие с АЦП такого типа требует реализации контроллера SPI в режиме ведомого. Предлагаемый СФ-блок состоит из ведомого контроллера SPI, блока конфигурационных регистров и модуля прямого доступа к памяти (ПДП). Использование ПДП позволяет минимизировать участие центрального процессора: последовательность данных, принятых от АЦП, накапливается во внутреннем буфере, после чего происходит пакетная передача по заданному адресу в системную память без программного вмешательства. Блок поддерживает работу с АЦП разрядностью 10, 12, 14 и 16 бит. В блоке реализованы однократный, кольцевой режимы ПДП и система прерываний по событиям заполнения буфера.

Ключевые слова: АЦП, SPI, ПДП, FIFO, SAR-преобразователи, SoC, AXI

1. Введение

При разработке систем на кристалле (SoC) возникает выбор, выполнять проектирование собственного АЦП или использовать внешний преобразователь. Использование внешнего АЦП позволяет избежать рисков применения непроверенного заказного блока и упростить топологию кристалла, но это решение требует периферийного блока приема данных в составе SoC.

Среди источников измерительных данных важное место занимают АЦП с последовательным интерфейсом. Они компактны и легко интегрируются в вычислительную систему. Для минимизации участия процессора в передаче данных целесообразно использовать АЦП в режиме ведущего. Такой АЦП самостоятельно формирует внутренний тактовый сигнал SCLK и выдает результат преобразования последовательным потоком битов. Подобное поведение характерно, например, для SAR-преобразователей типа ADS8513 [1]. В такой схеме принимающая сторона должна работать в режиме ведомого – принимать данные по внешнему тактовому сигналу, не иницируя обмен самостоятельно.

Чаще всего контролер SPI, реализованный в составе системного контроллера микропроцессора или микроконтроллера, выступает в роли задающего по шине SPI,

генерируя тактовый сигнал, однако для данной задачи такой контроллер неприменим.

Таким образом, актуальной является задача разработки компактного периферийного СФ-блока, объединяющего ведомый SPI контроллер, блок конфигурационных регистров и передачу результатов в память по каналу ПДП без остановки потока данных и без участия процессора в пересылке отдельных отсчетов.

В настоящей статье описаны архитектура блока, программная модель управляющих регистров, механизм упаковки данных и система прерываний. Работоспособность подтверждена функциональным моделированием при различных значениях разрядности входных данных.

2. Анализ известных решений

Современные системы сбора данных строятся на базе FPGA и систем на кристалле (SoC), совмещающих аппаратную обработку с программным управлением. Среди реализованных подходов можно выделить несколько характерных направлений.

Первое направление – комплексные SoC-платформы, в которых аппаратный прием данных совмещен с первичной обработкой непосредственно на кристалле. Такие решения применяются, в частности, в физических экспериментах. Так, для пиксельных детекторов предложена компактная система сбора данных на базе SoC FPGA со встроенным

микропроцессором и клиент-серверным приложением для удаленного управления и визуализации [2]. Авторы характеризуют ее как систему, которая «способна одновременно обслуживать несколько пиксельных детекторов». Близкое решение для позиционно-чувствительных кремниевых детекторов, по описанию авторов, «обеспечивает оцифровку сигналов детектора, хранение и передачу пакетированных данных на компьютер, обработку и визуализацию декодированной информации» [3]. Данные решения эффективны для конкретных установок, однако не рассчитаны на применение в виде самостоятельных сложнофункциональных блоков, которые можно интегрировать в проектируемые системы на кристалле.

Второе направление – высокопроизводительные многоканальные реализации на FPGA, где на первый план выходят пропускная способность и минимизация задержек. Так, в [4] отмечается, что сбор данных и их обработка, в существующих системах, выполняются на разных устройствах. Такой подход требует дополнительного времени на передачу, и в качестве решения авторы предлагают одночиповую структуру, в которой все процессы выполняются на одном кристалле [4]. По масштабу и сложности это существенно отличается от задачи одноканального СФ-блока.

Третье направление – гибкие системы с развитой логикой запуска и буферизации. Для регистрации быстрых переходных процессов созданы SoC-решения с возможностью быстрого повторного запуска и устойчивой работой при высокой интенсивности потока данных [5]. К этому же направлению относятся открытые FPGA-платформы сбора данных с SPI контроллерами и буферизацией в оперативной памяти, ориентированные на организацию

обратной связи в реальном времени [6].

Отдельную категорию составляют работы, посвященные унифицированным подсистемам передачи данных в память. ПДП-контроллер на шине AXI (Advanced eXtensible Interface – высокопроизводительный расширяемый интерфейс) позволяет существенно снизить нагрузку на процессор при непрерывном потоке и одинаково применим как в составе SoC, так и в системах с шиной PCIe [7]. Однако эти контроллеры являются самостоятельными блоками передачи и не включают в себя приемную периферию.

Таким образом, большинство рассмотренных решений ориентировано на крупные многоканальные платформы или законченные системы под конкретную задачу. Конфигурируемый блок, предназначенный для приема данных по SPI в режиме ведомого с последующей автономной записью в память, в опубликованных работах практически не встречается. Поэтому новизна предлагаемого подхода заключается не в самих принципах ведомого приема или ПДП-передачи, а в их архитектурной интеграции в единый компактный СФ-блок с конфигурируемой разрядностью принимаемых данных, их упаковкой и последующей записью в память с возможностью использования кольцевого буфера.

3. Предлагаемое решение

Решение задачи приема данных от АЦП, работающего в режиме задающего, без участия процессора было достигнуто архитектурным построением, основанным на разделении функций приема последовательного интерфейса и передачи данных по внутренней шине AXI.

Структурная схема разработанного IP-блока показана на рисунке 1.

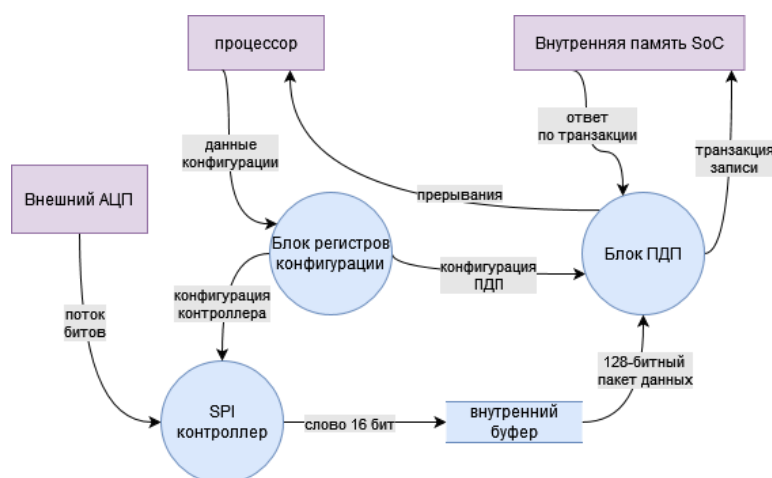


Рис. 1. Структурная схема СФ-блока

В состав блока входят: модуль ведомого SPI, конфигурационные регистры и модуль ПДП. Взаимодействие с системной памятью и регистрами управления осуществляется также через шину AXI. Настройка блока выполняется через набор конфигурационных регистров, с помощью которых можно задать параметры, такие как разрядность преобразователя, адрес буфера в памяти, размер передачи и параметры канала ПДП, в том числе маски прерываний и режим работы.

Логика движения данных организована следующим образом. На вход АЦП со стороны SoC подается сигнал запуска преобразования. В качестве такого сигнала может использоваться выход таймера, формирующий меандр заданной частоты. После запуска внешний аналого-цифровой преобразователь формирует тактовый сигнал SCLK и последовательный поток данных. Модуль контроллера захватывает передаваемые биты по фронту внешнего тактового сигнала, формирует из них слово разрядностью 16 бит. В случае если заданная разрядность АЦП меньше, контроллер добавляет ведущие нули в старших разрядах.

Из-за относительно небольшой разрядности преобразователей, данные одного считывания имеют ширину меньшую, чем ширина системной шины передачи, поэтому несколько последовательных отсчетов объединяют в более

широкое слово: восемь отсчетов упаковываются в 128-битный пакет. При такой организации уменьшается частота обращений к памяти, также это позволяет согласовать последовательный интерфейс преобразователя с широкой внутрисистемной шиной. После каждой последовательной передачи адрес буфера памяти инкрементируется на 16 байт.

Алгоритм работы блока реализован в виде конечного автомата, реализующего последовательность управления. После сброса, в начальном состоянии (IDLE) блок ожидает сигнал разрешения работы ПДП, который задается программно через конфигурационный регистр. После получения разрешения, блок переходит в состояние накопления отсчетов (ACCUMULATE), последовательно заполняя внутренний буфер. Накопив пакет данных шириной 128 бит, автомат инициирует запись в память (DMA_REQ) с помощью ПДП, ожидает ее завершения (DMA_WAIT) и после этого возвращается в начальное состояние. Если сигнал разрешения работы ПДП по-прежнему активен, в следующем такте блок сразу начинает накопление нового пакета. Блок-схема описанного алгоритма представлена на рисунке 2. Разрешение проверяется в начале каждого цикла, что позволяет корректно останавливать работу блока между пакетами, не теряя данные посередине передачи.

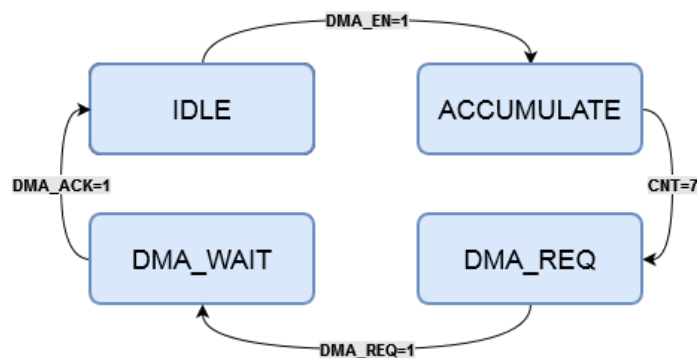


Рис. 2. Алгоритм работы блока

Блок ПДП поддерживает два режима работы, которые различаются поведением после завершения передачи: одиночный и кольцевой. В однократном режиме после завершения заданной передачи блок прекращает работу автоматически. В кольцевом режиме, после заполнения буфера памяти, следующая запись происходит по нулевому смещению, что может приводить к перезаписыванию старых данных. Чтобы избежать потери данных, предусмотрена генерация прерываний в двух случаях: по событию половинного и полного заполнения буфера, что позволяет программному обеспечению приступить к обработке данных

параллельно с их поступлением, а также не допустить перезаписывания данных до их использования.

Благодаря тому, что весь канал от приема последовательных данных до их записи в системную память работает самостоятельно, участие процессора минимально и сводится к начальному конфигурированию регистров и, при необходимости, обработке прерываний. Таким образом сбор данных обеспечивается без программного копирования данных.

Для проверки предложенной архитектуры была создана поведенческая модель АЦП на базе микросхемы ADS8513 [71]. Она

способна формировать последовательный поток данных, внутренний тактовый сигнал, а также изменять разрядность выходного слова от 10 до 16 бит с шагом 2 бита.

В ходе моделирования контролировались точность приема слова относительно эталона от модели АЦП, корректность упаковки

128-битного пакета из последовательных отсчетов, сохранность данных при записи в память, а также корректная работа канала ПДП в одиночном и кольцевом режиме. Процесс записи в память одного пакета данных показан на рисунке 3.

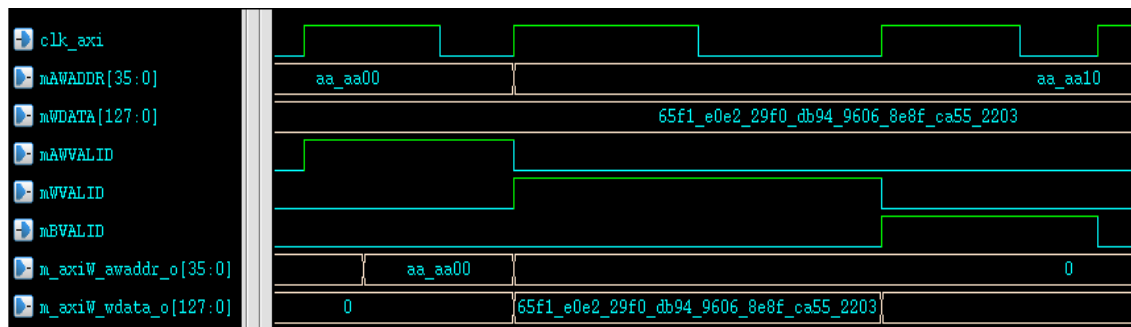


Рис. 3. Запись 128-битного пакета в память

5. Заключение

Разработан специализированный СФ-блок для приема данных от внешнего АЦП по интерфейсу SPI в режиме ведомого с автономной передачей в системную память по каналу ПДП.

Результаты моделирования подтвердили корректное функционирование блока, характеризующееся отсутствием потерь данных, записью по заданным адресам, циклическим возвратом адресации к началу буфера в кольцевом режиме ПДП и формированием прерываний в ожидаемые моменты. Наследующем этапе запланирована реализация прототипа в ПЛИС и проверка работоспособности на реальном АЦП.

В перспективных микроконтроллерах предложенное решение может послужить основой для реализации процесса получения данных от внешних микросхем, работающих в режиме SPI-ведущего. Дальнейшие исследования будут направлены на поддержку различных режимов тактирования SPI, организацию многоканального ввода и верификацию блока после топологического синтеза с аннотированными задержками, а также макетированию в ПЛИС.

Работа выполнена в рамках государственного задания НИЦ «Курчатовский институт» – НИИСИ по теме FNEF-2024-0003.

Data Acquisition Module for an External SPI ADC with DMA Support

E. A. Soloveva, A. V. Kornilenko

Abstract. An IP block has been developed that provides the reception of streaming data from an external analog-to-digital converter (ADC) over the SPI interface. A distinctive feature of this block is that the ADC acts as the master, generating the clock signal and controlling data exchange on the SPI bus. Interfacing with an ADC of this type requires implementing the SPI controller in slave mode. The proposed IP block consists of an SPI slave controller, a configuration register block, and a direct memory access (DMA) module. The use of DMA minimizes the involvement of the central processor: the sequence of data received from the ADC is accumulated in an internal buffer, after which it is transferred as a burst to a specified address in system memory without software intervention. The block supports operation with ADCs of 10, 12, 14, and 16 bits in resolution; single and circular DMA modes are also implemented, along with an interrupt system triggered by buffer-fill events.

Keywords: ADC, SPI, DMA, FIFO, SAR converters, SoC, AXI

Литература

1. Texas Instruments. ADS8513 16-bit, 40-kSPS Serial Output Analog-to-Digital Converter. Datasheet, SBAS112.
2. Jiménez Sánchez, J. Advanced System on Chip Field Programmable Gate Array Powered Data Acquisition System for Pixel Detectors / J. Jiménez Sánchez, P. Blanco Carmona, J. Hinojo [и др.] // *Sensors*. — 2023. — Vol. 24. — Art. 218. — DOI 10.3390/s24010218.
3. Topko, Yu. SoC FPGA Based Data Acquisition System for Position Sensitive Silicon Detectors / Yu. Topko, B. Topko, S. Khabarov, N. Zamyatin // *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. — 2022. — Т. 1033. — Ст. 166680. — DOI 10.1016/j.nima.2022.166680.
4. Choi, S. FPGA-Based Multi-Channel Real-Time Data Acquisition System / S. Choi, H. Yang, Y. Noh, G. Kim, E. Kwon, H. Yoo // *Electronics*. — 2024. — Vol. 13. — Art. 2950. — DOI 10.3390/electronics13152950.
5. Manduchi, G. A Versatile Board for Event Driven Data Acquisition / G. Manduchi, A. Rigoni Garola, L. Trevisan, T. Patton // *Sensors*. — 2024. — Т. 24. — Ст. 1631. — DOI 10.3390/s24051631.
6. Van Muoi, N. A Reproducible FPGA–ADC Synchronization Architecture for High Speed Data Acquisition / N. Van Muoi, N. Dong // *Data*. — 2026. — Т. 11. — Ст. 23. — DOI 10.3390/data11010023.
7. Zabolotny, W. Versatile DMA Engine for High Energy Physics Data Acquisition Implemented with High Level Synthesis / W. Zabolotny // *Electronics*. — 2023. — Т. 12. — Ст. 883. — DOI 10.3390/electronics12040883.

Поступила в редакцию / Received: 17.06.2026.

Поступила после рецензирования / Revised: 30.06.2026.

Принята к печати / Accepted: 30.06.2026.

УДК 004.32:004.312

Параметризация блока подключения приемопередатчиков контроллера PCI Express для оптимизации использования ресурсов ПЛИС и автоматизации верификации

Е.С. Зайцева¹, А.Д. Лыспак²¹НИЦ «Курчатовский институт» — НИИСИ, Москва, zaytseva@cs.niisi.ras.ru;²НИЦ «Курчатовский институт» — НИИСИ, Москва, alex_l@cs.niisi.ras.ru

Аннотация. В статье рассматривается задача подключения контроллера PCI Express разработки НИИСИ РАН к высокоскоростным приемопередатчикам FPGA семейства Xilinx UltraScale+. Исходная реализация контроллера использовала фиксированную конфигурацию приемопередатчиков на 16 линий независимо от фактической ширины канала PCI Express. Такой подход приводил к избыточному использованию аппаратных ресурсов ПЛИС при реализации конфигураций с меньшим числом линий. Предложен масштабируемый интерфейсный блок, обеспечивающий автоматический выбор конфигурации приемопередатчиков в зависимости от требуемой ширины канала PCI Express. Разработанное решение поддерживает режимы Endpoint и Root Complex, скорости Gen1–Gen3 и ширины канала x1, x2, x4, x8, x16. Для проверки работоспособности разработан параметризуемый скрипт запуска тестов, позволяющий выполнять моделирование любой из поддерживаемых конфигураций одной командой без ручного изменения структуры проекта. Экспериментальная оценка выполнена на плате Xilinx Virtex UltraScale+ VCU128. Полученные результаты показывают снижение аппаратных затрат при использовании конфигураций с меньшим числом линий и подтверждают эффективность автоматизированного подхода к верификации контроллера.

Ключевые слова: PCI Express, ПЛИС, GTY, SerDes, высокоскоростные интерфейсы, параметризация, автоматизация верификации

1. Введение

Интерфейс PCI Express (PCIe) является основным стандартом высокоскоростного взаимодействия между вычислительными устройствами и периферийными компонентами современных вычислительных систем [1–3]. Одной из ключевых особенностей PCIe является возможность масштабирования пропускной способности путем изменения количества используемых линий передачи данных (lane), что позволяет адаптировать интерфейс под конкретные требования.

Современные контроллеры PCIe реализуются как на специализированных микросхемах (ASIC), так и на программируемых логических интегральных схемах (ПЛИС), обеспечивающих гибкость проектирования и ускорение процесса разработки [4–6]. При этом физический уровень PCIe обычно реализуется с использованием встроенных высокоскоростных приемопередатчиков SerDes.

В известных FPGA-реализациях контроллеров PCIe основное внимание уделяется достижению максимальной пропускной способности интерфейса [3, 7, 8]. Однако при поддержке нескольких конфигураций PCIe возникают дополнительные задачи оптимизации аппаратных затрат и сокращения трудоемкости верификации.

Рост сложности цифровых устройств приводит к необходимости автоматизации процедур тестирования и аппаратной верификации FPGA-проектов [9–11]. Особенно актуальной данная задача становится для контроллеров высокоскоростных интерфейсов, поддерживающих большое количество конфигураций.

Целью работы является разработка масштабируемого интерфейсного блока подключения контроллера PCI Express к приемопередатчикам ПЛИС, обеспечивающего эффективное использование аппаратных ресурсов и автоматизацию процесса верификации.

2. Архитектура контроллера PCI Express и постановка задачи

Объектом исследования в данной работе является не сам контроллер PCI Express, а интерфейсный блок сопряжения, обеспечивающий его подключение к встроенным высокоскоростным приёмопередатчикам SerDes GTY FPGA семейства Xilinx UltraScale+. Именно этот модуль (см. рисунок 1) и его параметризация являются предметом настоящего исследования.

Используемый в работе контроллер PCI Express, разработанный в НИИСИ РАН, реализует функциональность уровней Transaction, Data Link и Physical в соответствии со спецификацией PCIe Gen3 и поддерживает работу в режимах Endpoint (EP) и Root Complex (RC), а также все основные ширины канала: x1, x2, x4, x8 и x16.

Таким образом, контроллер обеспечивает поддержку 30 конфигураций, определяемых комбинацией:

- 2 режимов (EP / RC);
- 3 скоростей (Gen1 / Gen2 / Gen3);
- 5 ширин канала (x1 / x2 / x4 / x8 / x16).

Задача интерфейсного модуля – автоматически выбирать конфигурацию GTY, соответствующую заданной ширине канала, обеспечивая масштабируемость и эффективное использование ресурсов ПЛИС.

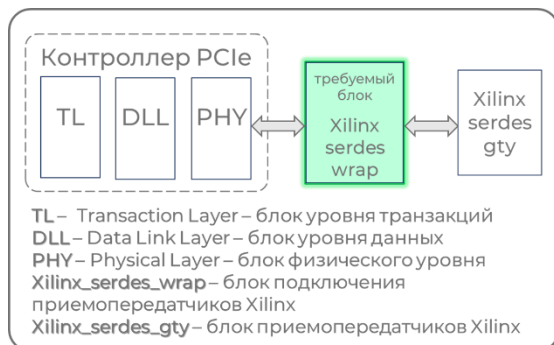


Рис. 1. Структурная схема подключения контроллера PCIe к приемопередатчикам Xilinx SerDes

3. Разработка масштабируемого интерфейсного блока

3.1. Исходная реализация и её недостатки

В первоначальной версии FPGA-проекта подключение контроллера PCI Express к приёмопередатчикам выполнялось через единственный, жёстко заданный экземпляр

IP-ядра GTY, сконфигурированный на максимальную ширину канала x16. Такой подход был выбран на этапе первичной отладки как наиболее универсальный, поскольку он гарантировал работоспособность контроллера во всех режимах, включая x16.

Однако при использовании контроллера в конфигурациях с меньшим числом линий (x1, x2, x4, x8) данный подход приводил к избыточному и неоправданному расходу аппаратных ресурсов ПЛИС: все 16 трансиверов оставались задействованными, независимо от реальной ширины канала PCIe. При этом значительная часть трансиверов простаивала, но продолжала занимать кристалл, а также приводила к избыточному энергопотреблению, что снижало эффективность использования FPGA в ресурсо-ограниченных проектах.

Таким образом, было сформулировано требование к новому решению: интерфейсный блок должен автоматически выбирать конфигурацию GTY, строго соответствующую требуемой ширине канала, и подключать ровно столько линий, сколько необходимо для работы контроллера в заданном режиме.

3.2. Предлагаемое решение: параметризуемый блок подключения

Суть предлагаемого решения заключается в создании отдельных конфигураций IP-ядер высокоскоростных приёмопередатчиков GTY для каждой поддерживаемой ширины канала (x1, x2, x4, x8, x16) и разработке параметризуемого интерфейсного модуля, который обеспечивает автоматический выбор и подключение именно того экземпляра GTY-ядра, который соответствует заданной ширине канала.

На первом этапе были сформированы специализированные IP-ядра GTY для конфигураций x1, x2, x4 и x8 (помимо уже имеющейся x16). Настройка каждого IP-ядра выполнялась в среде AMD Vivado с учетом требований стандарта PCI Express для режимов Gen1, Gen2 и Gen3.

На втором этапе был разработан адаптивный параметризуемый интерфейсный модуль, который по значению параметра PHY_LANE автоматически выбирает и подключает соответствующее IP-ядро (см. рисунок 2):

PHY_LANE = 1 → GTY x1;

PHY_LANE = 2 → GTY x2;

PHY_LANE = 4 → GTY x4;

PHY_LANE = 8 → GTY x8;

PHY_LANE = 16 → GTY x16.

Таким образом, вместо жёсткой привязки к x16 новое решение обеспечивает соответствие количества используемых приемопередатчиков требуемой конфигурации PCIe: для канала x8

задействуется ровно 8 линий, для x4 — 4 и т.д. Это позволяет отказаться от использования единственной фиксированной конфигурации x16 и обеспечить поддержку различных вариантов реализации контроллера в рамках единого проекта. Вследствие чего предоставляется возможность сократить занимаемые ресурсы, в частности, количество используемых высокоскоростных каналов GT, а также снизить потребляемую мощность.

```

generate
  if(PHY_LANE == 1)
    pcie_x1_gen1_gt pcie_gen1_gt_i
    (
      .port_list(port_list)
    );
  if(PHY_LANE == 2)
    pcie_x2_gen1_gt pcie_gen1_gt_i
    (
      .port_list(port_list)
    );
  if(PHY_LANE == 4)
    pcie_x4_gen1_gt pcie_gen1_gt_i
    (
      .port_list(port_list)
    );
  if(PHY_LANE == 8)
    pcie_x8_gen1_gt pcie_gen1_gt_i
    (
      .port_list(port_list)
    );
  if(PHY_LANE == 16)
    pcie_x16_gen1_gt pcie_gen1_gt_i
    (
      .port_list(port_list)
    );
endgenerate

```

Рис. 2. Подключение IP-ядер в общий проект (на примере реализации со скоростью Gen1)

4. Автоматизация верификации

Для проверки работоспособности всех поддерживаемых конфигураций разработан параметризуемый скрипт запуска тестов, позволяющий задавать режим работы (EP/RC), скорость (Gen1–Gen3) и ширину канала (x1–x16) одной командой, например:

./<скрипт_запуска> EP G1 X4.

Благодаря этому можно запустить конкретную требуемую конфигурацию без необходимости ручной перенастройки проекта, а также без прогона всех 30 вариантов. На основании переданных параметров автоматически выбирается конфигурация GTY, формируются параметры моделирования, запускается тест и собираются результаты. Такой подход исключает ручное изменение проекта для каждой конфигурации, снижает трудоёмкость подготовки тестирования и уменьшает вероятность ошибок конфигурирования.

5. Экспериментальные результаты

Для оценки эффективности предложенного решения была выполнена реализация контроллера PCI Express разработки НИИСИ РАН в ПЛИС для всех поддерживаемых конфигураций интерфейса: x1, x2, x4, x8 и x16. Экспериментальные исследования проводились на отладочной плате Xilinx Virtex UltraScale+ VCU128 с кристаллом XCVU37P-L2FSVH2892E. Для каждой конфигурации были выполнены синтез, размещение и трассировка (implementation) в среде Vivado, по результатам которых определены показатели использования аппаратных ресурсов ПЛИС (LUT, LUTRAM, FF, BRAM, DSP, IO, GT и BUFG), потребляемой мощности и времени реализации проекта.

В статье для сокращения объема для демонстрации предлагаемого подхода приведены результаты только для конфигураций Gen1. Аналогичные исследования выполнены также для конфигураций Gen2 и Gen3 и демонстрируют такой же характер изменения показателей. Результаты количественного сравнения использования аппаратных ресурсов для скорости Gen1 приведены в таблице 1.

Таблица 1. Сравнительная характеристика занимаемых аппаратных ресурсов ПЛИС XCVU37P-L2FSVH2892E для скорости Gen1 между исходной реализацией («было») и разработанной («стало»)

Аппаратные ресурсы ПЛИС	Было	Стало								
		Осталось	Добавленные конфигурации							
	x16	x16	x8		x4		x2		x1	
	Занято, шт.		Занято, шт.	Экономия по сравнению с x16 («было»), %	Занято, шт.	Экономия по сравнению с x16 («было»), %	Занято, шт.	Экономия по сравнению с x16 («было»), %	Занято, шт.	Экономия по сравнению с x16 («было»), %
GT-каналы	16		8	50	4	75	2	88	1	94
LUT	170720		91245	47	82291	52	77315	55	74976	56
LUTRAM	926		599	35	503	46	455	51	431	53
FF	93495		52204	44	45112	52	41170	56	39210	58
BRAM	45		23	49	23	49	23	49	23	49
BUFG	24		15	37	11	54	9	62	8	67

Результаты показывают, что использование специализированных конфигураций приемопередатчиков позволяет значительно снизить аппаратные затраты по сравнению

с исходной реализацией, основанной на фиксированной конфигурации x16. На рисунке 3 представлена сравнительная диаграмма для наглядности этого.

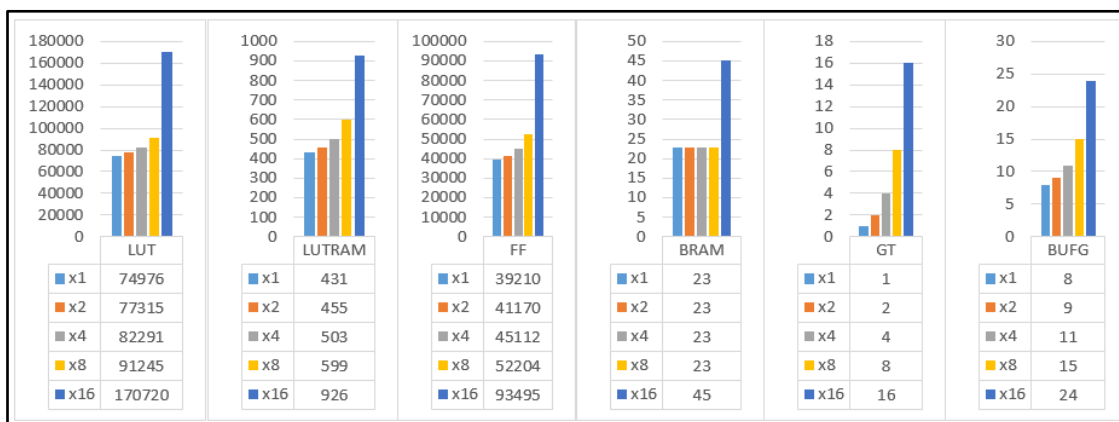


Рис. 3. Сравнительная диаграмма занимаемых ресурсов ПЛИС XC7VU37P-L2FSVH2892E для скорости Gen1 для конфигураций x1 – x16: снижение занятости при переходе от исходной фиксированной конфигурации x16 («было») к адаптивным конфигурациям x8, x4, x2 и x1 («стало»)

На рисунке 4 представлено сравнение совокупной мощности (Total On-Chip Power), рассеиваемой на кристалле ПЛИС и её составляющих, для различных конфигураций

шины PCIe (x1, x2, x4, x8, x16) при работе на скорости Gen1. Конкретные значения представлены в таблице 2.

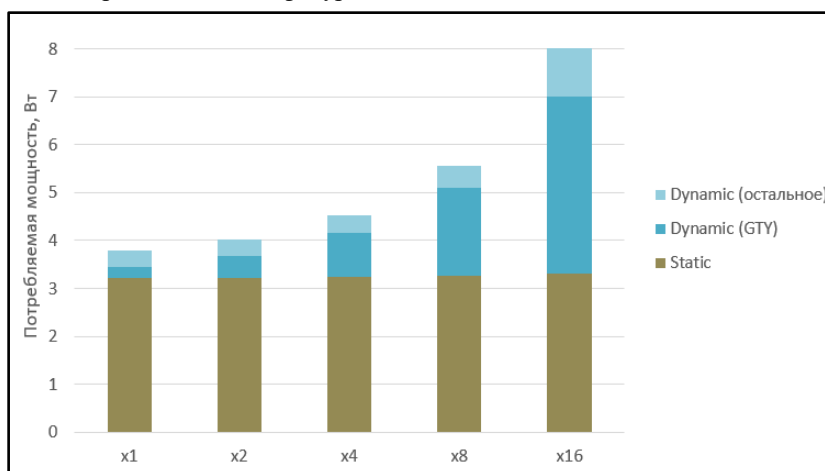


Рис. 4. Сравнительная диаграмма потребляемой мощности ПЛИС XC7VU37P-L2FSVH2892E для скорости Gen1 для конфигураций x1 – x16: значительная экономия потребления при переходе от исходной фиксированной конфигурации x16 («было») к адаптивным конфигурациям x8, x4, x2 и x1 («стало»)

Таблица 2. Сравнительная характеристика совокупной потребляемой мощности ПЛИС XC7VU37P-L2FSVH2892E и её составляющих для скорости Gen1 между исходной реализацией («было») и разработанной («стало»)

Потребляемая мощность	Было	Стало								
		Осталось		Добавленные конфигурации						
	x16	x16	x8		x4		x2		x1	
	Вт		Вт	Экономия по сравнению с x16 («было»), %	Вт	Экономия по сравнению с x16 («было»), %	Вт	Экономия по сравнению с x16 («было»), %	Вт	Экономия по сравнению с x16 («было»), %
Total On-Chip	8,012		5,550	31	4,530	43	4,034	50	3,784	53
– Static	3,316		3,259	2	3,236	2	3,225	3	3,219	3
– Dynamic	4,696		2,291	51	1,294	72	0,809	83	0,565	88
– GTU	3,699		1,849	50	0,925	75	0,462	88	0,231	94

Как видно из диаграммы, снижение количества используемых высокоскоростных линий закономерно приводит к значительному уменьшению динамической составляющей мощности (Dynamic Power), особенно большой вклад в эту составляющую вносят GT-каналы (GTY), в то время как статическая составляющая (Static Power) остаётся практически неизменной для всех конфигураций, что соответствует физической природе КМОП-структур.

Анализ результатов показывает, что уменьшение числа используемых линий PCI Express приводит к закономерному снижению аппаратных затрат, а вследствие чего и потребляемой мощности (в особенности, из-за сокращения динамической мощности), поскольку количество подключенных приемопередатчиков GTY и обслуживающей их логики теперь также уменьшается. Так на примере сравнения конфигурации x1 с исходной x16 наблюдается следующий выигрыш в занимаемых ресурсах ПЛИС (для скорости Gen1):

- GT — уменьшается в 16 раз (на 94%);
- LUT — уменьшается на 56%;
- LUTRAM — уменьшается на 53%;
- FF — уменьшается на 58%;
- BRAM — уменьшается на 49%;
- BUFG — уменьшается на 67%.

Вследствие чего наблюдается следующий выигрыш в потребляемой мощности (тоже для конфигурации x1 относительно исходной x16 и для скорости Gen1):

- статическая составляющая мощности (Static) — уменьшается незначительно (на 3%);
- динамическая составляющая мощности (Dynamic) — значительно сокращается (на 88%), огромный вклад в её сокращение вносит уменьшение мощности, потребляемой GT (сокращается на 94%); на 66% также уменьшается динамическая мощность, потребляемая остальной логикой (без учета GT-трансиверов) в связи с уменьшением других используемых аппаратных ресурсов;

– и, таким образом, общая мощность (Total On-Chip), состоящая из статической и динамической, — сокращается на 53%.

Кроме того, время синтеза, размещения и трассировки проекта сокращается до 2 раз при использовании конфигураций с меньшим

числом линий (x1–x8) по сравнению с исходной реализацией x16, что особенно важно при частых итерациях проектирования и отладки.

Таким образом, предложенный подход обеспечивает рациональное использование ресурсов ПЛИС в зависимости от требуемой конфигурации контроллера PCI Express, снижение потребляемой мощности, поддержку 30 вариантов аппаратной реализации на основе единой RTL-архитектуры и сокращает время синтеза, размещения и трассировки проекта, а также обеспечивает автоматизацию подготовки и выполнения функциональной верификации.

6. Заключение

В работе разработан параметризуемый интерфейсный модуль подключения контроллера PCI Express к высокоскоростным приемопередатчикам GTY ПЛИС семейства Xilinx UltraScale+. Предложенное решение обеспечивает автоматический выбор конфигурации GTY (x1, x2, x4, x8, x16) в зависимости от требуемой ширины канала, поддерживая режимы Gen1–Gen3, Endpoint и Root Complex на основе единой RTL-реализации.

Экспериментальные результаты, полученные на плате VCU128, подтверждают эффективность подхода: при переходе от фиксированной конфигурации x16 к параметризуемой x1 число задействованных GTY-каналов сокращается в 16 раз, занятость LUT — на 56 %, FF — на 58 %, динамическая мощность — на 88 %, а общая мощность на кристалле — на 53 %. Время синтеза и размещения проекта сокращается до 2 раз.

Разработан также параметризуемый скрипт для автоматизированной верификации всех 30 поддерживаемых конфигураций, позволяющий запускать тесты одной командой без ручной перенастройки проекта, что снижает трудоёмкость тестирования и уменьшает вероятность ошибок конфигурирования.

Таким образом, предложенный подход обеспечивает масштабируемость, эффективное использование ресурсов ПЛИС и автоматизацию верификации, что уменьшает трудоёмкость разработки и сопровождения контроллеров высокоскоростных интерфейсов.

Parametrization of the transceiver connection block of a PCI Express controller for FPGA resource optimization and verification automation

E. S. Zaytseva, A. D. Lyspak

Abstract. This paper addresses the integration of a PCI Express controller developed at NIISI RAS with high-speed transceivers of Xilinx UltraScale+ FPGAs. The initial implementation used a fixed x16 transceiver configuration regardless of the actual PCI Express link width, leading to excessive hardware resource utilization when operating in narrower-lane modes. A scalable interface block is proposed that automatically selects the appropriate transceiver configuration based on the required PCIe link width. The developed solution supports Endpoint and Root Complex modes, Gen1–Gen3 data rates, and x1, x2, x4, x8, x16 lane widths. To verify functionality, a parameterizable testbench script has been developed, allowing simulation of any supported configuration with a single command without manual project restructuring. Experimental evaluation was performed on the Xilinx Virtex UltraScale+ VCU128 board. The obtained results demonstrate hardware savings for narrower-lane configurations and confirm the effectiveness of the proposed automated verification approach.

Keywords: PCI Express, FPGA, GTY, SerDes, high-speed interfaces, parametrization, verification automation

Литература

1. PCI-SIG. PCI Express Base Specification Revision 3.0. PCI Special Interest Group, 2010.
2. Shanley T., Anderson D. PCI Express System Architecture. Addison-Wesley Professional, 2009.
3. MindShare Inc. PCI Express Technology 3.0. MindShare Press, 2012.
4. AMD. UltraScale Architecture GTY Transceivers User Guide UG578. 2024.
5. AMD. Vivado Design Suite User Guide: Designing with IP UG896. 2024.
6. Li H., Liu Y., Yuan D., Hu H. A Wrapper of PCI Express with FIFO Interfaces Based on FPGA // arXiv:1306.5501, 2013.
7. Bittner R. Bus Mastering PCI Express in an FPGA // Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays. 2009. P. 145–150.
8. Cheng K., Liu W., Shen Q., Liao S. Design and Implementation of High-throughput PCIe with DMA Architecture between FPGA and PowerPC // arXiv:1809.07702, 2018.
9. Степченко Ю.А., Морозов Н.В., Дьяченко Ю.Г. и др. Аппаратная верификация рекуррентного обработчика сигналов на ПЛИС // Проблемы разработки перспективных микро- и нанoeлектронных систем. 2021.
10. Гаврилов С.В., Железников Д.А., Заплетина М.А. и др. Разработка доверенных средств проектирования цифровых схем на базе гетерогенных ПЛИС // Труды ИСП РАН. 2023. Т. 35. № 6. С. 61–76.
11. Mizuhiki R., Maeda J., Marumoto S. Development of the Firmware Logic Validation System Using FPGA Accelerator // Journal of Instrumentation. 2025.

Поступила в редакцию / Received: 03.07.2026.

Поступила после рецензирования / Revised: 09.07.2026.

Принята к печати / Accepted: 10.07.2026.

УДК 004.9:621.01

Параметрический синтез допустимого базового профиля винтовых роторов SRM на основе генетического алгоритма

А. А. Полин¹, В. А. Горелов¹, В. Ж. Куклин¹, А. Ю. Сафронов¹

¹НИИЦ «Курчатовский институт» – НИИСИ, Москва, Российская Федерация, polin.alex@yandex.com

Аннотация. Описана процедура автоматизации подбора и построения профилей винтовых роторов типа SRM (Svenska Rotor Maskiner). В работе рассматривается классический случай винтовых роторов с постоянным по длине шагом винтовой линии и неизменной формой поперечного сечения профиля от окна всасывания до окна нагнетания. Предложен метод параметрического синтеза базового профиля – исходного профиля, служащего отправной точкой для последующей оптимизации его формы и удовлетворяющего техническому заданию и условиям геометрической допустимости (далее – допустимого базового профиля, или кратко базового профиля) по техническому заданию, в котором поиск девяти проектных параметров выполняется генетическим алгоритмом с контролем геометрической и кинематической допустимости с параллельной оценкой особей. Корректность синтезированной геометрии подтверждается независимым расчётом объёма парной полости двумя методами. На тестовом задании для ступени сжатия попутного нефтяного газа допустимый профиль получен за несколько поколений при наибольшей частной погрешности соответствия техническому заданию около 4,8 %; расхождение объёма по двум методам не превысило 0,5 %. Метод реализован как сквозной автоматизированный цикл с выгрузкой координат профиля для построения в системах CAD.

Ключевые слова: винтовой компрессор, профиль ротора SRM, генетический алгоритм, автоматизированное проектирование, параметрический синтез, допустимость геометрии

1. Введение

Винтовые компрессоры применяются в нефтегазовой, химической, холодильной и энергетической отраслях благодаря компактности, непрерывной подаче и способности работать в широком диапазоне расходов и давлений. Их энергетическая эффективность во многом закладывается на этапе профилирования: форма сопряжённых профилей ведущего и ведомого роторов задаёт характер изменения объёма парной полости, геометрию окон всасывания и нагнетания, длину линий контакта и потенциальные пути перетечек [1, 2]. Поэтому проектирование профиля – самостоятельная инженерная задача, в которой геометрические, кинематические и термодинамические требования приходится согласовывать одновременно.

Традиционно эта задача решается полуручными методами. Базовые параметры роторов подбираются по справочным методикам и опыту конструктора, а изменение объёма рабочей полости определяется графоаналитически [3]. Такой путь оправдан для единичного расчёта, но плохо масштабируется: подбор параметров под новое техническое задание требует множества согласованных пересчётов, зависит от квалификации

исполнителя и слабо приспособлен к быстрой проработке целых линеек профилей.

Современные работы по винтовым компрессорам смещают акцент в сторону численного моделирования и оптимизации [4]. Разработаны трёхмерные модели рабочего процесса [5, 6] и постановки оптимизации профиля по параметрам зацепления [7, 8], а в смежных инженерных задачах хорошо зарекомендовали себя эволюционные и многокритериальные методы поиска [9, 10]. Отдельное направление – методы оптимизации формы и топологии конструкций [11, 12]. Прямой перенос топологической оптимизации на профиль винтового ротора, однако, затруднён: произвольное изменение связности области нарушает сопряжённость пары, поэтому для данной задачи обоснованнее параметризованная оптимизация формы при сохранении допустимости.

При всей проработанности отдельных методов между ними остаётся разрыв. Подбор базовых параметров, построение сопряжённых профилей, расчёт рабочих полостей и подготовка геометрии для CAD обычно выполняются разрозненными средствами и требуют ручного связывания. Единого автоматизированного цикла, который по техническому заданию выдавал бы

геометрически и кинематически допустимый профиль, пригодный для дальнейшего анализа и передачи в CAD, в инженерной практике недостаёт; отечественных программных решений такого рода, по сведениям авторов, не обнаружено.

В настоящей работе предложен метод автоматизированного параметрического синтеза базового профиля винтовых роторов типа SRM по заданным условиям работы компрессора. Поиск проектных параметров выполняется генетическим алгоритмом, в целевую функцию которого встроен контроль допустимости профиля, а корректность построенной геометрии подтверждается независимым расчётом объёма парной полости двумя методами с последующей выгрузкой координат профиля в формат, пригодный для импорта в CAD. Проведенный авторами библиографический и патентный поиск показал, что в открытых научных публикациях, патентной литературе и доступных источниках не выявлено отечественных программных средств для автоматизированного параметрического синтеза профилей винтовых роторов типа SRM в рассматриваемой постановке задачи. В частности, подобные программные решения не упоминаются в современном учебном пособии по компрессорной технике [13]. Это определяет практическую ценность предложенного подхода как основы для создания отечественного инженерного программного обеспечения.

Цель работы – разработать метод и алгоритмическое обеспечение автоматизированного синтеза допустимых профилей винтовых роторов SRM по техническому заданию. Практическая значимость состоит в снижении влияния человеческого фактора при подборе параметров, ускорении проработки новых линеек профилей и создании основы для отечественного инженерного программного обеспечения проектирования винтовых роторов.

2. Метод автоматизированного синтеза профиля

2.1. Постановка задачи и исходные данные тестового расчета

Исходными данными служит техническое задание на компрессорную ступень. В рассматриваемом тестовом расчёте задавались: давление и температура всасывания 0,103 МПа и 323 К; давление и температура нагнетания 1,55 МПа и 373 К; объёмная производительность 6983 м³/ч; частота вращения ведущего ротора 1500 об/мин;

молярная масса газа 40,91 г/моль, плотность 1,569 кг/м³, показатель адиабаты 1,154 и газовая постоянная 203,2 Дж/(кг·К). По этим данным программа вычисляет массовый расход, внешнюю степень сжатия и показатель политропы и выбирает типоразмер ступени – внешний диаметр D_1 и относительную длину Kl – по номограмме подбора маслозаполненного компрессора [14].

Результатом тестового расчета должен быть вектор варьируемых параметров, который воспроизводит заданный режим работы с приемлемой погрешностью и задаёт геометрически и кинематически допустимый базовый профиль. Совокупность требований к точности воспроизведения режима работы и геометрической реализуемости базового профиля определяет негладкую, частично дискретную допустимую область, для поиска в которой применён генетический алгоритм.

Близость особи к заданному режиму оценивается через частные погрешности. Для каждого контролируемого параметра технического задания частная погрешность – это относительное отклонение расчётного значения от заданного:

$$\delta_i = |y_i - y_i^{T3}| / |y_i^{T3}| \cdot 100\%, \quad (1)$$

где δ_i – частная погрешность по i -му параметру, %; y_i – расчётное значение i -го параметра; y_i^{T3} – значение i -го параметра по техническому заданию; $i = 1, 2, \dots, n$; n – число контролируемых параметров.

Наибольшей частной погрешностью соответствия техническому заданию называется максимальная из них по всем контролируемым параметрам:

$$\varepsilon = \max_i \delta_i, \quad (2)$$

где δ_i – частная погрешность по i -му параметру, %; ε – наибольшая частная погрешность соответствия техническому заданию, %.

Она показывает, насколько от задания отклоняется худший из параметров. По совокупности частных погрешностей с учётом штрафов за нарушение геометро-кинематической допустимости формируется целевая функция, минимизируемая алгоритмом:

$$F(x) = \sum_{i=1}^n \delta_i(x) + P(x), \quad (3)$$

где $F(x)$ – целевая функция особи x ; $\delta_i(x)$ – частные погрешности по (1); n – число контролируемых параметров; $P(x)$ – штрафная составляющая, равная нулю для геометрически и кинематически допустимой особи, добавляющая положительный вклад за каждое нарушенное условие допустимости и принимающая заведомо большое значение при грубой недопустимости.

Значение целевой функции (3) служит мерой приспособленности особи: по нему особи ранжируются и отбираются генетическим алгоритмом, причём меньшему значению отвечает более приспособленная особь (постановка на минимизацию). Поэтому «лучшей» считается особь с меньшим значением целевой функции, а «улучшением» – снижение этого значения от поколения к поколению; наибольшая частная погрешность ϵ по (2) приводится дополнительно как наглядный показатель качества. Все контролируемые параметры технического задания (производительность, степень сжатия, окружная скорость) строго положительны, а их допустимые диапазоны не содержат нуля, поэтому нормирование частной погрешности на заданное значение корректно.

2.2. Кодирование решения

Каждое решение (особь) кодируется вектором из девяти нормированных генов $g_i \in [0, 1]$ ($i = 1, \dots, 9$). Декодирование – линейное отображение каждого гена в свой диапазон из таблицы 1 – превращает гены в варьируемые параметры. По варьируемым параметрам рассчитываются производные геометрические величины (начальные диаметры, внешний диаметр, межосевое расстояние, длина роторов, ход винтовой линии), после чего строится сопряжённый профиль. Нормировка генов к единому отрезку делает работу операторов скрещивания и мутации независимой от физических масштабов варьируемых параметров.

2.3. Диапазоны варьируемых параметров и проверка допустимости

Девять варьируемых параметров изменяются в пределах диапазонов, рекомендованных в литературе по расчёту винтовых компрессоров [1, 2, 5]; приведённые в таблице 1 границы отвечают практике маслозаполненных винтовых компрессоров, на данных по которым основаны эти источники. Эти диапазоны задают границы, в которые декодируются нормированные гены (таблица 1). Поскольку нормированные гены декодируются именно в эти диапазоны, варьируемые параметры всегда остаются в технологически осмысленных пределах.

По варьируемым параметрам рассчитаны производные геометрические величины: начальные диаметры роторов, внешний диаметр и межосевое расстояние. В отличие от варьируемых параметров, которые по построению лежат в своих диапазонах, производные величины при неудачном сочетании генов могут выйти за физические допустимые пределы, поэтому каждая из них

дополнительно проверяется. Если рассчитанное значение не укладывается в свой диапазон – например, внешний диаметр превышает предельный для выбранного типоразмера или нарушается условие сопряжённости начальных диаметров, – особь признаётся недопустимой и получает заведомо большое значение целевой функции, фактически исключая её из дальнейшего поиска.

Таблица 1. Диапазоны варьируемых параметров

Параметр	Обозн.	Диапазон
Изотермический КПД	$\eta_{из}$	0,60–0,80
Коэффициент подачи	λ	0,85–0,92
Окружная скорость ведущего ротора, м/с	u_1	20–55
Механический КПД	$\eta_{мех}$	0,90–0,96
Коэффициент качества профиля ведущего ротора	χ_1	0,30–0,50
Коэффициент качества профиля ведомого ротора	χ_2	0,40–0,60
Отн. высота головки зуба ведущего ротора	ξ_1	0,55–0,65
Отн. высота головки зуба ведомого ротора	ξ_2	0,060–0,065
Угол закрутки ведущего ротора, град	τ_1	260–320

2.4. Операторы и адаптивная настройка алгоритма

Алгоритм реализован средствами библиотеки DEAP [15]. Начальная популяция из 300 особей формируется случайно в пределах диапазонов варьируемых параметров (таблица 1). На каждом поколении выполняется цикл «отбор – скрещивание – мутация – формирование нового поколения».

Отбор родителей проводится турнирной селекцией с размером турнира 5: из случайно выбранных пяти особей в родители проходит лучшая. Поскольку недопустимые особи получают заведомо большое значение целевой функции за счет штрафной составляющей по (3), в турнире они практически всегда уступают допустимым и почти допустимым решениям и не проходят в родители. Тем самым селекция автоматически вытесняет недопустимые особи из размножения, не удаляя их из популяции жёстко: ограничения учитываются методом штрафов непосредственно через целевую функцию. Скрещивание, смешивающее (BLX- α) с $\alpha = 0,5$: гены потомка выбираются из интервала, расширенного за пределы родительских значений. Мутация – полиномиальная с контролем границ; её резкость задаётся параметром распределения η , а доля изменяемых генов – вероятностью мутации. Применяется элитизм: 10 % лучших особей (не менее пяти) копируются в следующее

поколение без изменений, что гарантирует, что лучшее найденное решение не ухудшается между поколениями.

Настройки алгоритма изменяются по ходу поиска по линейному расписанию в зависимости от относительного прогресса. Вероятность скрещивания снижается с 0,7 примерно до 0,49, вероятность мутации – с 0,4 до нижнего предела 0,1, а параметр η полиномиальной мутации – с 50 до 25. Смысл расписания – сместить баланс от широкого исследования пространства решений на ранних поколениях к уточнению найденных решений на поздних. Чтобы поиск не терял перспективные решения у границы допустимой области, отдельно отслеживаются лучшее допустимое и лучшее почти допустимое решение. Предусмотрена ранняя остановка по стагнации.

2.5. Параллельная реализация и вычислительный конвейер

Оценка приспособленности – вычисление значения целевой функции (3) для особи – наиболее затратная операция: для каждой особи строится сопряжённый профиль и выполняется геометрический анализ рабочих полостей по углу поворота. Поэтому особи поколения оцениваются параллельно: расчёт распределяется по процессам с числом рабочих, равным меньшему из 10 и числа логических процессоров системы, $p = \min(10, N_{\text{лог}})$. Эксперименты проводились на персональной рабочей станции с процессором AMD Ryzen 5 5600H (6 физических ядер, 12 логических процессоров), 32 ГБ оперативной памяти, под управлением Windows 10; для этой конфигурации $p = 10$. Это серийный персональный компьютер общего назначения; все вычисления выполнялись на центральном процессоре, графические ускорители и иные специализированные вычислительные модули не использовались. Для расчётной нагрузки, ограниченной производительностью процессора, ускорение близко к линейному до числа физических ядер, а при большем числе процессов дополнительный выигрыш даёт одновременная многopotочность (SMT), но уже с убывающей отдачей. Верхняя граница в 10 процессов (из 12 логических) выбрана как компромисс между временем прогона и нагрузкой на процессор: она существенно сокращает время оценки популяции из 300 особей и при этом оставляет часть логических процессоров свободными, благодаря чему рабочая станция сохраняет отзывчивость в течение всего расчёта. Библиотека DEAP поддерживает распараллеливание и на многопроцессорных установках и кластерах через MPI, однако в настоящей работе оно не

применялось; перенос оценки особей на кластер с MPI рассматривается как направление дальнейшего сокращения времени расчёта. Метод оформлен как сквозной автоматизированный цикл (рис. 1): техническое задание → генетический поиск параметров → построение профилей и корпусов → анализ полостей и окон → проверка корректности → выгрузка координат профиля для импорта в CAD. Внутренняя логика генетического поиска показана на рис. 2.

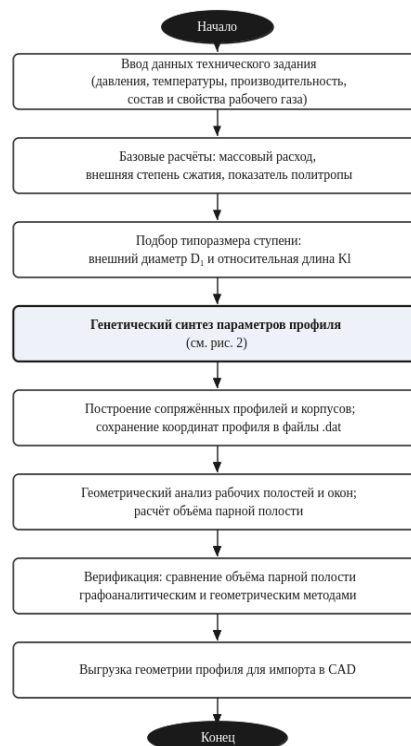


Рис. 1. Процессная схема автоматизированного синтеза профиля

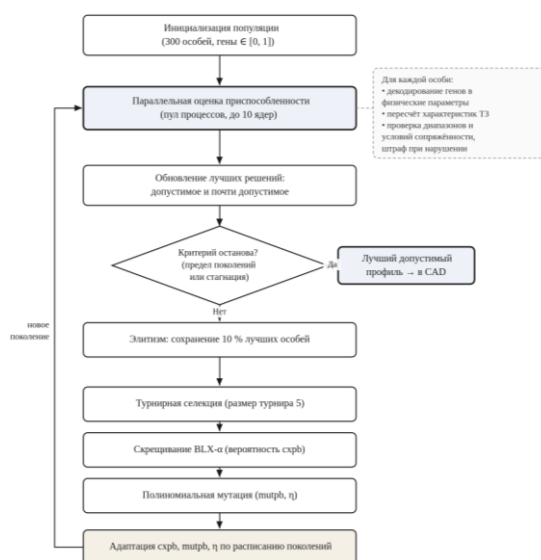


Рис. 2. Цикл работы генетического алгоритма

3. Вычислительный эксперимент

3.1. Условия эксперимента

Метод проверен на синтезе профиля ступени для сжатия попутного нефтяного газа по техническому заданию из раздела 2.1. По введённым данным программа рассчитала массовый расход 10 956,33 кг/ч, внешнюю степень сжатия 15,05 и показатель политропы 1,06, после чего выбрала типоразмер ступени: внешний диаметр $D_1 = 510$ мм и относительную длину $Kl = 1,3$. Генетический поиск выполнялся при популяции 300 особей; оценка особей распараллеливалась на 10 рабочих процессах. Полное время прогона составило около одного часа.

3.2. Сходимость алгоритма

Динамика поиска приведена в таблице 2 и на рис. 3а–3в. Уже в начальной популяции присутствовало почти допустимое решение с наибольшей частной погрешностью 12,3 %, не удовлетворявшее, однако, всем условиям допустимости. По мере смены поколений суммарная погрешность лучшего решения монотонно снижалась ($158,9 \rightarrow 95,9 \rightarrow 56,6 \rightarrow 12,5$), и на пятом поколении был найден первый полностью допустимый профиль с наибольшей частной погрешностью 4,76 %. Наиболее резкое улучшение приходится на четвёртое и пятое поколения, когда удачные комбинации генов закрепляются отбором и элитизмом и быстро распространяются по популяции.

не удовлетворявшее, однако, всем условиям допустимости. По мере смены поколений суммарная погрешность лучшего решения монотонно снижалась ($158,9 \rightarrow 95,9 \rightarrow 56,6 \rightarrow 12,5$), и на пятом поколении был найден первый полностью допустимый профиль с наибольшей частной погрешностью 4,76 %. Наиболее резкое улучшение приходится на четвёртое и пятое поколения, когда удачные комбинации генов закрепляются отбором и элитизмом и быстро распространяются по популяции.

Таблица 2. Динамика эволюции лучшего решения по поколениям

Поколение	Статус	Сумм. погр.	Макс. частн. погр., %
0	почти допуст.	158,9	12,28
1	почти допуст.	158,9	12,28
2	почти допуст.	95,9	9,05
3	почти допуст.	56,6	6,60
4	почти допуст.	12,5	5,15
5	допустимое	8,96	4,76

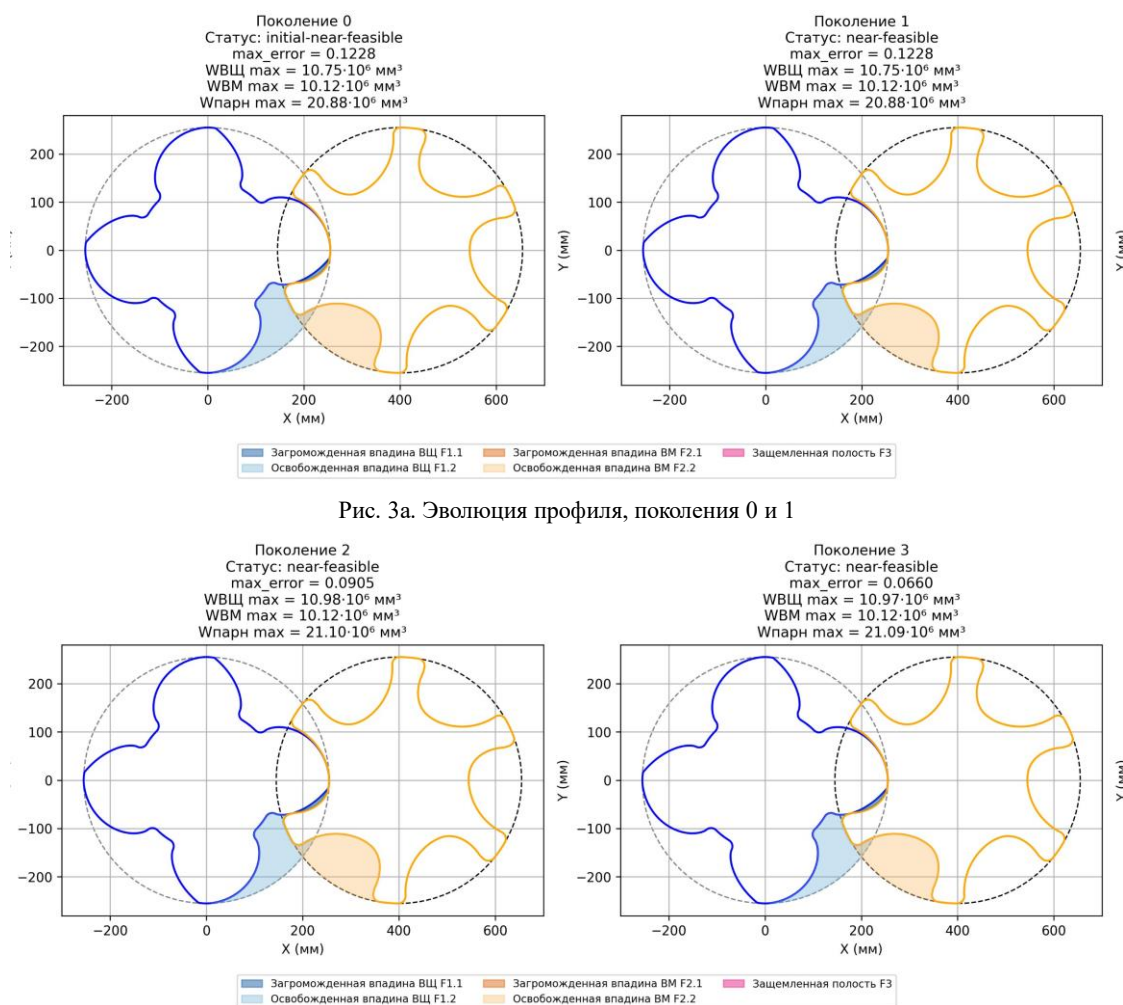


Рис. 3а. Эволюция профиля, поколения 0 и 1

Рис. 3б. Эволюция профиля, поколения 2 и 3

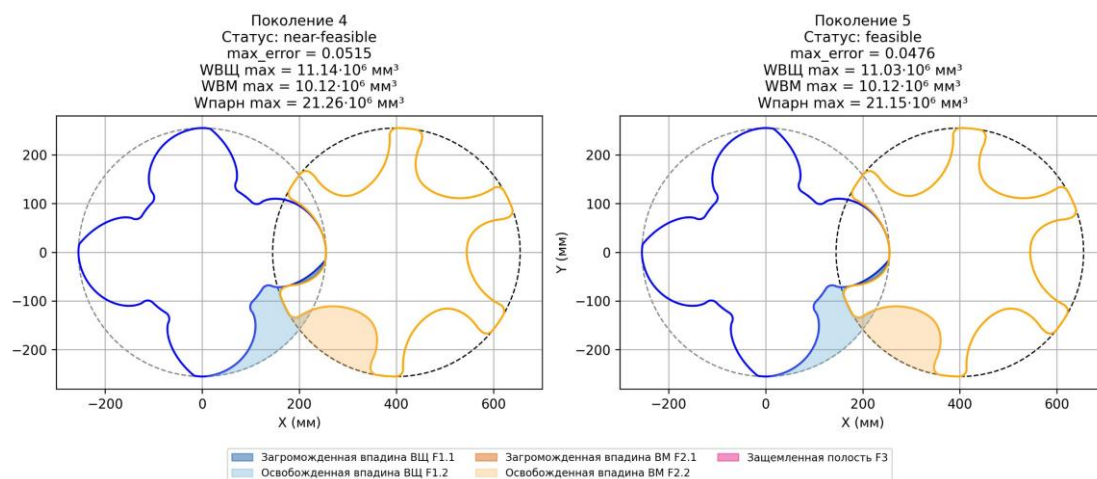
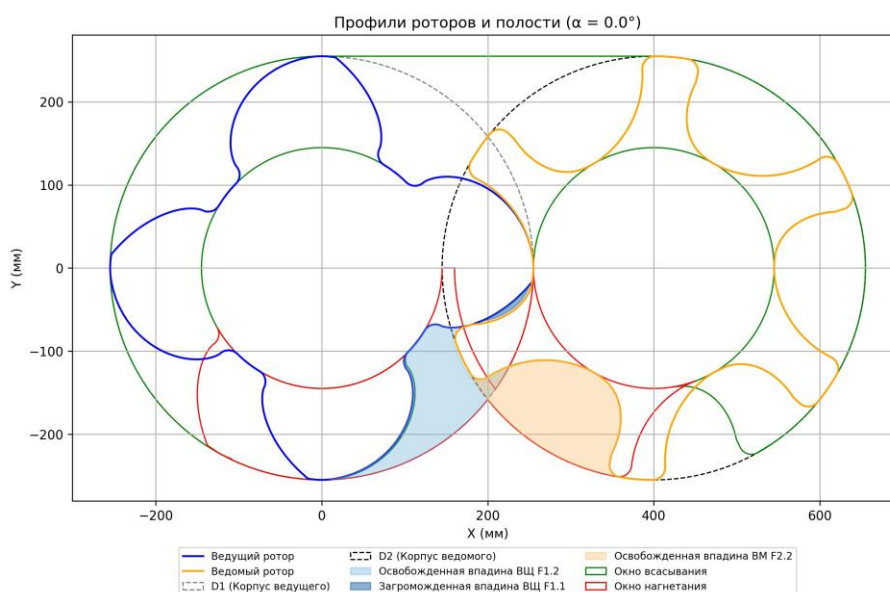


Рис. 3в. Эволюция профиля, поколения 4 и 5

3.3. Синтезированный профиль

Найденное допустимое решение задаёт сопряжённую пару роторов с числами зубьев $z_1 = 4$ и $z_2 = 6$, внешним диаметром $D_1 = D_2 = 510$ мм, межосевым расстоянием $A = 400$ мм и длиной роторов $L = 663$ мм; рассчитанная объёмная производительность совпадает с заданной с точностью долей процента. Помимо профилей, программа автоматически

строит окна всасывания и нагнетания, угловые размеры которых определяются по результатам анализа изменения объёма рабочих полостей. Профили роторов, образованные ими рабочие полости и построенные окна всасывания и нагнетания при нулевом угле поворота показаны на рис. 4. Координаты точек профиля автоматически сохраняются в файлы формата .dat, по которым строится кривая профиля в CAD-системе.

Рис. 4. Профили роторов, рабочие полости и окна всасывания и нагнетания при $\alpha = 0^\circ$

3.4. Самопроверка результата

Корректность построенной геометрии подтверждается независимым расчётом объёма парной полости двумя методами – графоаналитическим и геометрическим (рис. 5). Начальный рабочий объём, полученный геометрическим методом ($21\,151\,902\text{ мм}^3$), отличается от значения графоаналитического

метода ($21\,048\,038\text{ мм}^3$) на $0,49\%$, а от теоретического максимального объёма – на $1,67\%$. Согласованность полученных независимых оценок подтверждает геометрическую непротиворечивость синтезированного профиля. Заданная геометрическая степень сжатия $4,50$ достигается при угле нагнетания $253,35^\circ$ (расчётное значение составляет $4,47$).

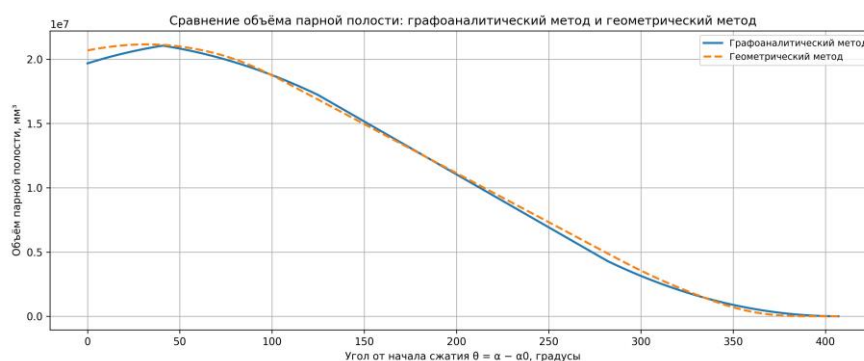


Рис. 5. Сравнение объёма парной полости: графоаналитический и геометрический методы

4. Обсуждение

Результаты эксперимента подтверждают, что предложенный подход позволяет получать базовый профиль автоматически и воспроизводимо. Поиск, начавшись с почти допустимых решений, за несколько поколений выходит в область, где одновременно удовлетворяются требования технического задания и условия геометро-кинематической допустимости, а наибольшая частная погрешность снижается до уровня, приемлемого для предварительного проектирования.

Главное достоинство метода — перевод трудоёмкого полуручного подбора параметров в управляемый автоматизированный цикл. Это снижает влияние квалификации и субъективных решений исполнителя на результат, ускоряет проработку вариантов и позволяет за один прогон получать согласованный набор геометрии вместе со средствами её проверки. Раздельное отслеживание допустимых и почти допустимых решений и адаптивное расписание операторов удерживают поиск у границы допустимой области и повышают шанс найти рабочее решение при жёстких ограничениях.

Метод имеет естественные границы применимости. Поскольку метод оперирует только геометрией и кинематикой профиля и не моделирует рабочую среду, наличие или отсутствие впрыска масла на сам алгоритм синтеза не влияет — он одинаково применим кмаслозаполненным компрессорам и к машинам сухого сжатия. Привязка к типу компрессора возникает лишь через числовые диапазоны варьируемых параметров (таблица 1), которые в настоящей работе взяты из источников по маслозаполненным винтовым компрессорам [1, 2, 5]. Для машин сухого сжатия, отличающихся, в частности, повышенными окружными скоростями и меньшими степенями повышения давления в ступени, эти диапазоны следует заменить на характерные для данного класса; методика и алгоритм при этом остаются

прежними. На текущем этапе целевая функция оценивает соответствие техническому заданию и геометрическую допустимость, но не физическую эффективность: КПД, перетечки, энергопотребление и шумовые характеристики напрямую не оптимизируются. Модель остаётся геометрической и не учитывает термогазодинамику рабочего процесса; вычислительная стоимость одного синтеза составляет около часа на указанной рабочей станции. Поэтому полученный профиль следует рассматривать как обоснованное начальное приближение для последующего анализа.

Развитие работы видится в переходе от синтеза базового профиля к многокритериальной оптимизации его формы по физическим показателям с привлечением суррогатных моделей [16] и верификацией методами вычислительной гидрогазодинамики и конечных элементов. Проведенный библиографический и патентный поиск показал, что в открытых научных публикациях, патентной литературе и доступных источниках не выявлено отечественных программных средств, предназначенных для автоматизированного параметрического синтеза профилей винтовых роторов типа SRM в рассматриваемой постановке задачи, что определяет практическую ценность предложенного подхода как основы для инженерного программного обеспечения.

5. Заключение

Предложены метод и алгоритмическое обеспечение автоматизированного параметрического синтеза допустимых профилей винтовых роторов типа SRM по техническому заданию. Поиск проектных параметров выполняется генетическим алгоритмом с встроенным контролем геометро-кинематической допустимости, раздельным отслеживанием допустимых и почти допустимых решений, адаптивной настройкой

операторов и параллельной оценкой особенностей; построенная геометрия проверяется независимым расчётом объёма парной полости двумя методами и выгружается в формат, пригодный для импорта в CAD.

На тестовом задании для ступени сжатия попутного нефтяного газа допустимый профиль получен за несколько поколений при наибольшей частной погрешности соответствия заданию около 4,8 %, а расхождение объёма парной полости по двум методам не превысило 0,5 %. Тем самым подтверждена работоспособность метода как сквозного автоматизированного цикла.

Практическая значимость работы состоит в снижении влияния человеческого фактора при подборе параметров, ускорении проработки новых профилей и создании основы для отечественного инженерного программного обеспечения проектирования винтовых роторов.

Дальнейшая работа будет направлена на многокритериальную оптимизацию формы профиля по физическим критериям.

Основные представленные результаты были получены в рамках подготовки диссертационной работы Полина А.А.

Parametric synthesis of admissible basic SRM screw-rotor profile based on a genetic algorithm

A. A. Polin, V. A. Gorelov, V. Zh. Kuklin, A. Yu. Safronov

Abstract. The automation of selecting and constructing SRM (Svenska Rotor Maskiner) screw-rotor profiles is considered. The study deals with the classical case of screw rotors having a constant helix lead along the rotor length and an unchanged cross-sectional profile from the suction port to the discharge port. A method is proposed for the parametric synthesis of a base profile, i.e., an initial profile serving as the starting point for its subsequent shape optimization and satisfying the technical specification and the conditions of geometric feasibility (hereinafter referred to as an admissible base profile, or simply a base profile). The search over nine design parameters is performed by a genetic algorithm with built-in control of geometric and kinematic feasibility and parallel evaluation of individuals. The correctness of the synthesized geometry is confirmed by an independent computation of the paired-cavity volume by two methods. For a test specification of an associated-petroleum-gas compression stage, a base profile was obtained within several generations with a maximum partial error of specification matching of about 4.8%; the discrepancy between the two volume-computation methods did not exceed 0.5%. The method is implemented as an end-to-end automated pipeline with export of profile coordinates for construction in CAD systems.

Keywords: screw compressor, SRM rotor profile, genetic algorithm, automated design, parametric synthesis, geometric feasibility

Литература

1. И.Г. Хисамеев, В.А. Максимов. Двухроторные винтовые и прямозубые компрессоры: теория, расчёт и проектирование. Казань, ФЭН, 2000.
2. L. Rinder. Schraubenverdichter. Wien, New York, Springer-Verlag, 1979.
3. А.В. Братусь, И.В. Автономова. Метод определения объёма парной полости винтового компрессора. «Известия высших учебных заведений. Машиностроение», 2015, № 7, 69–77. DOI: 10.18698/0536-1044-2015-7-11-19. URL: <https://izvuzmash.bmstu.ru/articles/1183/1183.pdf> (дата обращения: 14.05.2026).
4. Y. Li, X. Zhao, S. Liu, C. Wang, S. Shen, Y. Guo. Review and prospects of numerical simulation research on internal flow and performance optimization of twin-screw compressors. "Energies", V. 18 (2025), № 10, 2608. DOI: 10.3390/en18102608. URL: <https://doi.org/10.3390/en18102608> (дата обращения: 14.05.2026).
5. N. Stosic, I.K. Smith, A. Kovacevic. Screw Compressors: Mathematical Modelling and Performance Calculation. Berlin, Heidelberg, Springer, 2005.
6. F. Kaufmann, L. Irrgang, C. Schiffler, H. Spliethoff. Fast and accurate modelling of twin-screw compressors: a generalised low-order approach. "Applied Thermal Engineering", V. 257 (2024), 124238. DOI: 10.1016/j.applthermaleng.2024.124238. URL: <https://doi.org/10.1016/j.applthermaleng.2024.124238> (дата обращения: 14.05.2026).
7. K. Ilie, A. Subic. Parametric modelling of helical rotors for efficient design of twin-screw

superchargers. "Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science", V. 221 (2007), № 2, 267–272. DOI: 10.1243/0954406JMES421FT. URL: <https://journals.sagepub.com/doi/10.1243/0954406JMES421FT> (дата обращения: 14.05.2026).

8. A. Aydın, T. Engin, A. Kovacevic. Design optimisation of a screw compressor with a focus on rotor depth: a CFD approach. "International Journal of Refrigeration", V. 170 (2025), 385–397. DOI: 10.1016/j.ijrefrig.2024.12.001. URL: <https://doi.org/10.1016/j.ijrefrig.2024.12.001> (дата обращения: 14.05.2026).

9. D.E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, Addison-Wesley, 1989.

10. D.W. Zingg, M. Nemec, T.H. Pulliam. A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization. "European Journal of Computational Mechanics", V. 17 (2008), № 1–2, 103–126. DOI: 10.3166/remn.17.103-126. URL: <https://doi.org/10.3166/remn.17.103-126> (дата обращения: 14.05.2026).

11. M.P. Bendsoe, O. Sigmund. Topology Optimization: Theory, Methods and Applications. Berlin, Heidelberg, Springer, 2003.

12. G. Allaire, F. Jouve, A.-M. Toader. A level-set method for shape optimization. "Comptes Rendus Mathématique", V. 334 (2002), № 12, 1125–1130. DOI: 10.1016/S1631-073X(02)02412-3. URL: [https://doi.org/10.1016/S1631-073X\(02\)02412-3](https://doi.org/10.1016/S1631-073X(02)02412-3) (дата обращения: 14.05.2026).

13. И.В. Автономова. Компрессорные станции и установки газового комплекса: учебное пособие. Москва, Изд-во МГТУ им. Н.Э. Баумана, 2025. 200 с. ISBN 978-5-7038-6566-8.

14. I.M. Arbon. The Design and Application of Rotary Twin-shaft Compressors in the Oil and Gas Process Industry. London, Mechanical Engineering Publications, 1994.

15. F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, C. Gagne. DEAP: Evolutionary Algorithms Made Easy. "Journal of Machine Learning Research", V. 13 (2012), 2171–2175. URL: <https://www.jmlr.org/papers/v13/fortin12a.html> (дата обращения: 14.05.2026).

16. J.A. Manson, T.W. Chamberlain, R.A. Bourne. MVMOO: Mixed variable multi-objective optimisation. "Journal of Global Optimization", V. 80 (2021), 865–886. DOI: 10.1007/s10898-021-01052-9. URL: <https://link.springer.com/article/10.1007/s10898-021-01052-9> (дата обращения: 14.05.2026).

Поступила в редакцию / Received: 15.06.2026.

Поступила после рецензирования / Revised: 28.06.2026.

Принята к печати / Accepted: 01.07.2026.

УДК 004.65:004.91

Применение генератора отчетов для преобразования таблиц из формата текстового файла в формат электронной таблицы

А. Б. Бетелин¹¹НИЦ «Курчатовский институт» – НИИСИ, Москва, Россия, ab@niisi.msk.ru

Аннотация. В статье приводится описание методики преобразования таблиц из формата текстового файла в формат электронной таблицы при помощи генератора отчетов АСУ НИИСИ.

Ключевые слова: преобразование текстовых таблиц, электронная таблица, табличный процессор, генератор отчетов

1. Введение

В учетно-управленческой деятельности организации иногда возникает потребность преобразовать файл, содержащий таблицу с некоторыми данными, из формата простого текстового файла (plain text) в формат табличного процессора типа OpenOffice Calc [1] или Microsoft Excel [2].

Такая ситуация может сложиться, к примеру, когда в организации применяются "древние" программы, ориентированные исключительно на алфавитно-цифровое (а/ц) взаимодействие с пользователем. И если подобная программа порождает данные в виде таблицы, то в ее оформлении используются только алфавитно-цифровые знаки из присутствующих на клавиатуре а/ц терминала и еще, возможно, символы псевдографики.

Дело может осложняться необходимостью провести дополнительную переработку таблицы, например: удалить ряд колонок, добавить несколько новых, вычислив значения полей в этих колонках, изменить заголовки колонок и всей таблицы. Вдобавок, с целью повышения удобства визуального восприятия результата имеет смысл добавить внешнее оформление (например, цвет фона для ячеек), задать гарнитуры и кегли шрифтов, скорректировать ширину колонок и т.п. Т.е. кроме изменения формата еще и выполнить существенное преобразование структуры таблицы и содержащейся в ней информации.

Если подобное действие происходит достаточно редко или вообще однократно, то проще всего проделать его вручную средствами самого табличного процессора: загрузить в него

требуемый текстовый файл, отредактировать и сохранить в нужном формате.

Если же это действие регулярное, а содержимое обрабатываемых файлов однотипно, то имеет смысл автоматизировать этот процесс и осуществлять его в пакетном режиме, используя возможности командной строки.

Покажем, как такая задача может быть решена с применением генератора отчетов, разработанного в НИИСИ РАН (ныне НИЦ «Курчатовский институт» – НИИСИ - НИИСИ) для АСУ НИИСИ [3].

2. Пошаговое описание методики

Предлагаемая методика предполагает получение преобразованной таблицы в виде файла, порождаемого генератором отчетов АСУ НИИСИ (он подробно описан в [4]).

Для того, чтобы воспользоваться этим генератором отчетов, нужно:

а) сформировать входной поток информации для генератора;

б) создать шаблон отчета в формате ODS.

Входной поток необходимо получить из исходного текстового файла с таблицей, которую требуется подвергнуть преобразованиям.

Шаблон может быть изготовлен при помощи табличного процессора OpenOffice Calc или аналогичного ему. В шаблоне задаются внешнее оформление отчета и переменные, которые впоследствии будут заменены на конкретные значения из входного потока.

Поскольку мы предполагаем воспользоваться возможностями, предоставляемыми АСУ НИИСИ, для

формирования входного потока будет удобно загрузить исходный текстовый файл с таблицей в базу данных (БД), применяемую в АСУ НИИСИ, в ней средствами СУБД провести все необходимые преобразования таблицы и затем изготовить входной поток в виде CSV-файла.

АСУ НИИСИ функционирует в среде ОС Linux [5], поэтому мы будем пользоваться в том числе средствами командного интерпретатора bash [6].

Опишем по шагам последовательность действий для решения нашей задачи.

Будем считать, что программа генератора отчетов размещена в каталоге `/bin`, а все действия мы проводим в специально созданном для этого рабочем каталоге с именем `/wrk`. При этом предполагаем, что у нас имеются права на запись в этот каталог.

При помощи любого подходящего текстового редактора создадим специализированный командный файл на диалекте языка SQL для сервера СУБД MySQL [7], который эксплуатируется в АСУ НИИСИ. В этом файле запрограммируем загрузку исходного файла в БД, все необходимые преобразования данных исходной таблицы, а также команды подготовки входного потока информации для генератора отчетов. Назовем этот файл `tab.sql` и разместим его в каталоге `/wrk`.

Шаблон, содержащий образец внешнего вида отчета, также разместим в каталоге `/wrk` в файле с именем `tab.ods`. Сделаем каталог `/wrk` текущим (здесь и далее приводятся команды для командного интерпретатора bash):

```
cd /wrk
```

Изготовим входной поток в виде CSV-файла для генератора отчетов:

```
mysql -hlocalhost -u<username> \  
-p<userpasswd> -B <DBname> < tab.sql >  
tab.csv
```

Здесь:

<username> - имя пользователя СУБД MySQL;
<userpasswd> - пароль пользователя СУБД MySQL;
<DBname> - имя базы данных.

Далее необходимо выделить и переработать содержательную часть шаблона отчета - файл `content.xml`. Для этого выполним следующее.

В каталоге `/wrk` создадим временный каталог с именем `tmp`:

```
mkdir /wrk/tmp
```

Скопируем наш шаблон в этот каталог и там разархивируем его при помощи утилиты `unzip`:

```
cp /wrk/tab.ods /wrk/tmp  
cd /wrk/tmp && unzip tab.ods
```

После этого в каталоге `/wrk/tmp` должны появиться файлы из zip-архива шаблона, и среди них - файл `content.xml`, который необходимо модифицировать в процессе изготовления отчета. Для этого запустим генератор отчетов (программу `xmlfilter.pl` - см. [4]) следующим образом:

```
cat /wrk/tab.csv | /bin/xmlfilter.pl \  
--in /wrk/tmp/content.xml \  
--out /wrk/tmp/content.xml
```

Мы вправе указать один и тот же файл в качестве входного и выходного для генератора отчетов, т.к. сначала генератор полностью считывает входной файл и только после этого формирует выходной файл.

В результате в каталоге `/wrk/tmp` появится переработанный файл `content.xml` для готового отчета. Остальные файлы изменениям не подвергаются. Осталось упаковать все эти файлы в ODS-документ, который заменит шаблон отчета в каталоге `/wrk/tmp`:

```
cd /wrk/tmp && zip -r tab.ods *
```

Поместим полученный отчет в наш рабочий каталог `/wrk` под именем `tab_res.ods` и удалим уже ненужный временный каталог `/wrk/tmp`:

```
cp /wrk/tmp/tab.ods /wrk/tab_res.ods  
cd /wrk && rm -rf /wrk/tmp
```

В конечном итоге в каталоге `/wrk` образуется готовый отчет в виде ODS-документа, который содержит таблицу, подвергшуюся всем требуемым преобразованиям.

Если же имеется необходимость получить файл в формате электронной таблицы MS Excel, то можно сконвертировать наш ODS-документ в этот формат средствами пакета OpenOffice, пользуясь языком OpenOffice Basic.

Запустим OpenOffice в диалоговом режиме, затем последовательно откроем пункты меню "Сервис" -> "Макросы" -> "Управление макросами" -> "OpenOffice.org Basic". В открывшемся окне в каталоге объектов выберем "Мои макросы" -> "Standard" -> "Module1". Нажав кнопку "Редактировать", в модуль "Module1" согласно рекомендациям [8,9] добавим одну процедуру и одну функцию (в OpenOffice Basic и те и другие называются макросами):

Код:

```
sub To_MS_Excel ( File_name )
  File_URL = ConvertToURL ( File_name )
  Obj_doc = StarDesktop.loadComponentFromURL _
( File_URL, "_blank", 0, Array ( MakePropertyValue _
( "Hidden", True ), ) )
  if not isNull ( Obj_doc )
  then
    select case LCase ( Right ( File_name, 3 ) )
      case "ods"
        Flt_name = "Microsoft Excel 97"
        File_ext = "xls"
      case else
        File_ext = "000"
    end select
    File_name = Left ( File_name, _
Len ( File_name ) - 4 ) + "." + File_ext
    File_URL = ConvertToURL ( File_name )
    Obj_doc.storeToURL ( File_URL, Array _
( MakePropertyValue ( "FilterName", Flt_name ), ) )
    Obj_doc.close ( True )
  endif
end sub
```

```
function MakePropertyValue _
( optional Prop_name as String, optional Prop_value ) _
as com.sun.star.beans.PropertyValue
  dim ob_Struct as new _
com.sun.star.beans.PropertyValue
  if not IsMissing ( Prop_name )
  then
    ob_Struct.Name = Prop_name
  endif
  if not IsMissing ( Prop_value )
  then
    ob_Struct.Value = Prop_value
  endif
  MakePropertyValue () = ob_Struct
end function
```

Процедура To_MS_Excel выясняет расширение входного файла и если оно "ods", то организует конвертацию этого файла в формат MS Excel. Вспомогательная функция MakePropertyValue необходима для работы механизмов пакета OpenOffice. Она задает соответствие между указанным свойством (property) и его значением, и возвращает пару «свойство-значение» в виде структуры. Эта функция заранее не определена, поэтому ее нужно добавить самостоятельно.

Теперь можно вызвать OpenOffice в режиме командной строки, указав абсолютное имя входного файла в качестве аргумента процедуры To_MS_Excel:

```
soffice -headless -norestore -invisible \
-nofirststartwizard \
"macro:///Standard.Module1.To_MS_Excel \
(/wrk/tab_res.ods)"
```

Если приведенную выше пошаговую последовательность действий оформить в виде скрипта для командного интерпретатора bash, то мы получим средство автоматического преобразования таблиц из формата текстового файла в формат электронной таблицы MS Excel в пакетном режиме.

3. Дополнительные соображения

Функциональность процедуры To_MS_Excel может быть расширена, с тем чтобы она выполняла конвертацию и других типов файлов, обрабатываемых программами пакета OpenOffice - ODP и ODT. Добавим в процедуру To_MS_Excel необходимые строки кода (см. [8,9]) и для соответствия выполняемым операциям переименуем ее в To_MS_Office:

Код:

```
sub To_MS_Office ( File_name )
  File_URL = ConvertToURL ( File_name )
  Obj_doc = StarDesktop.loadComponentFromURL _
( File_URL, "_blank", 0, Array ( MakePropertyValue _
( "Hidden", True ), ) )
  if not isNull ( Obj_doc )
  then
    select case LCase ( Right ( File_name, 3 ) )
      case "odt"
        Flt_name = "Microsoft Word 97"
        File_ext = "doc"
      case "ods"
        Flt_name = "Microsoft Excel 97"
        File_ext = "xls"
      case "odp"
        Flt_name = "Microsoft PowerPoint 97"
        File_ext = "ppt"
      case else
        File_ext = "000"
    end select
    File_name = Left ( File_name, _
Len ( File_name ) - 4 ) + "." + File_ext
    File_URL = ConvertToURL ( File_name )
    Obj_doc.storeToURL ( File_URL, Array _
( MakePropertyValue ( "FilterName", Flt_name ), ) )
    Obj_doc.close ( True )
  endif
end sub
```

Теперь процедура To_MS_Office сможет обеспечить конвертацию файлов текстовых

документов, электронных таблиц и презентаций из форматов пакета OpenOffice в соответствующие форматы пакета Microsoft Office [10].

Также следует отметить, что требуемая модификация исходной таблицы может быть реализована не только с помощью средств, предоставляемых применяемым диалектом языка SQL. Для этого можно воспользоваться любым подходящим для этих целей языком программирования, например, Perl [11]. Или же, в зависимости от объема и сложности преобразований, провести обработку исходного файла (частично или полностью) при помощи утилит командной строки, предназначенных для операций с текстом, таких как `awk`, `sed`, `cut` [12] и т.п. Здесь также возможно использование вспомогательных программ обработки текста

`tes`, `tbe`, `pfm` и др., входящих в состав программного комплекса на основе редактора текстов RK [4,13]. Конечным результатом применения перечисленных средств должно стать формирование входного потока информации для генератора отчетов.

4. Заключение

Приведенная выше методика хорошо вписывается в сценарии автоматической обработки файлов. Она с успехом применяется в НИИСИ для регулярного преобразования текстовых файлов, содержащих таблицы с данными, в формат табличного процессора Microsoft Excel.

Using a report generator to convert tables from a text file format to a spreadsheet format

A. B. Betelin

Abstract. The article describes a technique for converting tables from a text file format to an electronic spreadsheet format using the report generator of the Automated Control System of SRISA.

Keywords: text tables converting, spreadsheet, spreadsheet processor, report generator

Литература

1. OpenOffice.org [Электронный ресурс] // Wikipedia. — URL: <https://en.wikipedia.org/wiki/OpenOffice.org> (дата обращения: 25.06.2026).
2. Microsoft Excel [Электронный ресурс] // Wikipedia. — URL: https://en.wikipedia.org/wiki/Microsoft_Excel (дата обращения: 25.06.2026).
3. Бетелин, А. Б. Методы создания распределённого комплекса информационных систем для автоматизации управленческого и бухгалтерского учёта / А. Б. Бетелин, И. Б. Егорычев, А. А. Прилипко, Г. А. Прилипко, С. Г. Романюк, Д. В. Самборский // Труды НИИСИ РАН. — 2018. — Т. 8, № 4. — С. 175–180.
4. Бетелин, А. Б. Простые генераторы отчётов для учётно-управленческих информационных систем / А. Б. Бетелин, И. Б. Егорычев, А. А. Прилипко, Г. А. Прилипко, С. Г. Романюк, Д. В. Самборский // Труды НИИСИ РАН. — 2023. — Т. 13, № 1–2. — С. 10–18.
5. Linux [Электронный ресурс] // Wikipedia. — URL: <https://en.wikipedia.org/wiki/Linux> (дата обращения: 25.06.2026).
6. Bash (Unix shell) [Электронный ресурс] // Wikipedia. — URL: [https://en.wikipedia.org/wiki/Bash_\(Unix_shell\)](https://en.wikipedia.org/wiki/Bash_(Unix_shell)) (дата обращения: 25.06.2026).
7. MySQL [Электронный ресурс] // Wikipedia. — URL: <https://en.wikipedia.org/wiki/MySQL> (дата обращения: 25.06.2026).
8. DuCharme, Bob. Moving to OpenOffice: Batch Converting Legacy Documents / Bob DuCharme. — 11.01.2006. — URL: <https://www.xml.com/pub/a/2006/01/11/from-microsoft-to-openoffice.html> (дата обращения: 25.06.2026).
9. Массовое преобразование файлов формата Microsoft Office в открытый формат Open Document из командной строки [Электронный ресурс]. — URL: <https://ulspo.ru/index.php/instructions/how-to/131-msf2odf-macros.html> (дата обращения: 25.06.2026).

10. Microsoft Office [Электронный ресурс] // Wikipedia. — URL: https://en.wikipedia.org/wiki/Microsoft_Office (дата обращения: 25.06.2026).
11. Perl [Электронный ресурс] // Wikipedia. — URL: <https://en.wikipedia.org/wiki/Perl> (дата обращения: 25.06.2026).
12. Тейнсли, Д. Linux и UNIX: программирование в shell. Руководство разработчика / Д. Тейнсли; пер. с англ. — Киев : Издательская группа BHV, 2001. — 464 с.
13. Прилипко, Г. А. Исследование и разработка интерактивных языково-ориентированных систем редактирования и визуализации программ: автореф. дис. ... канд. физ.-мат. наук / Г. А. Прилипко ; МГУ им. М. В. Ломоносова. — Москва, 1986. — 23 с.

Поступила в редакцию / Received: 19.02.2026.

Поступила после рецензирования / Revised: 26.06.2026.

Принята к печати / Accepted: 26.06.2026.

Наименование: сетевой рецензируемый научный журнал «Труды НИИСИ»

ISSN: 3033-6422

Сведения о переименовании: до 2025 года журнал издавался в печатном виде с названием «Труды НИИСИ РАН», ISSN 2225-7349

Журнал основан: 2011 г.

Периодичность: 4 раза в год

Учредитель и издатель: НИЦ «Курчатовский институт» — НИИСИ

Главный редактор: Бетелин Владимир Борисович, д. ф.-м. н., профессор, академик РАН

Адрес учредителя и издателя: 117218, Москва, Нахимовский проспект, д.36, к.1

Адрес редакции: 117218, Москва, Нахимовский проспект, д.36, к.1

Контакты: Тел.: +7 (925) 924-83-46; muranov@niisi.msk.ru

Подписка: Электронная версия журнала находится в свободном доступе на сайте журнала, а также в базах данных открытого доступа

Title: *SRISA Proceedings* online peer-reviewed journal

ISSN: 3033-6422

Former title: until 2025, the journal was published in print under the name *Trudy NIISI RAN* (Proceedings of SRISA, Russian Academy of Sciences), ISSN 2225-7349

Published since: 2011

Publication frequency: Quarterly

Founder and publisher: NRC "Kurchatov Institute" - SRISA

Chief Editor: Vladimir B. Betelin, Doctor of Science (Phys&Math), Prof., member of the Russian Academy of Sciences,

Address of the founder and publisher: 117218, Moscow, Nakhimovsky avenue, 36, bldg. 1

Address of the editorial office: 117218, Moscow, Nakhimovsky avenue, 36, bldg. 1

Contacts: Tel.: +7 (925) 924-83-46; muranov@niisi.msk.ru

Subscription: The electronic version of the journal is freely available on the journal's website, as well as in open access databases

Подписано в печать 13.07.2026 г.

Формат 60x90/8

Печать цифровая. Условных печатных листов – 8,4