DOI 10.37882/2223-2966.2025.03.09

УПРАВЛЕНИЕ АППАРАТНЫМИ РЕСУРСАМИ ОПЕРАЦИОННОЙ СИСТЕМЫ НА ОСНОВЕ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМ

MANAGEMENT OF HARDWARE RESOURCES OF THE OPERATING SYSTEM BASED ON LOGICAL INTEGRATED CIRCUITS

R. Valiev
I. Golovin
N. Ponomarev
A. Abdyusheva

Summary. This paper considers the problem of increasing the efficiency of high-performance computing systems due to the close integration of the operating system (OS) and logical integrated circuits (LIC). Modern approaches to the use of FPGA and ASIC as hardware accelerators are analyzed, examples of effective organization of interaction between the OS kernel and hardware modules are given. The authors propose a unified OS architecture capable of directly managing LIS resources, as well as a set of algorithms that optimize the distribution of computing resources under high loads and response time constraints. Based on theoretical comparisons, the advantage of the proposed solution over traditional systems, which are mainly focused on software resource management mechanisms, is shown.

Keywords: operating system, logic integrated circuits, FPGA, ASIC, hardware resources, high-performance computing, resource allocation, real-time control.

Валиев Руслан Ринатович
начальник отдела, Институт Нефтепереработки и Нефтехимии, ФГБОУ ВО УГНТУ, г. Салават
ruli811@mail.ru
Головин Игорь Николаевич
Институт Нефтепереработки и Нефтехимии, ФГБОУ ВО УГНТУ, г. Салават
golovin-igor-nikolaevich@mail.ru
Пономарёв Никита Александрович
Институт Нефтепереработки и Нефтехимии, ФГБОУ ВО УГНТУ, г. Салават
nikita.ponomaryov2022@yandex.ru
Абдюшева Алёна Денисовна
Институт Нефтепереработки и Нефтехимии, ФГБОУ ВО УГНТУ, г. Салават
alenaabduseva01@yandex.ru

Аннотация. В данной работе рассматривается проблема повышения эффективности высокопроизводительных вычислительных систем за счёт тесной интеграции операционной системы (ОС) и логических интегральных схем (ЛИС). Анализируются современные подходы к использованию FPGA и ASIC в качестве аппаратных ускорителей, приводятся примеры эффективной организации взаимодействия ядра ОС с аппаратными модулями. Авторами предложена унифицированная архитектура ОС, способная напрямую управлять ресурсами ЛИС, а также набор алгоритмов, позволяющих оптимизировать распределение вычислительных ресурсов в условиях высоких нагрузок и ограничений по времени отклика. На основе теоретических сравнений показывается преимущество предлагаемого решения перед традиционными системами, ориентирующимися, в основном, на программные механизмы управления ресурсами.

Ключевые слова: операционная система, логические интегральные схемы, FPGA, ASIC, аппаратные ресурсы, высокопроизводительные вычисления, распределение ресурсов, управление в реальном времени.

Современные тенденции в сфере высокопроизводительных вычислительных систем указывают на стремительный рост объёмов обрабатываемых данных и повышение требований к пропускной способности систем. Традиционно для ускорения вычислительных операций использовались либо более мощные универсальные центральные процессоры (CPU), либо специализированные микросхемы (GPU, DSP), однако в последнее десятилетие на первый план выходят логические интегральные схемы (ЛИС) — FPGA (Field-Programmable Gate Array) и ASIC (Application-Specific Integrated Circuit). Эти устройства обеспечивают эффективное выполнение широкого спектра задач при минимальных задержках и высоком уровне параллелизма [1–2].

Однако даже при наличии аппаратных ускорителей эффективность вычислений во многом определяется операционной системой, которая управляет распределением ресурсов, обрабатывает прерывания, обеспечивает механизм переключения контекстов и т. д. Во многих современных ОС взаимодействие с ЛИС носит частичный характер: специализированные драйверы, отдельные модули ввода-вывода (I/O) либо сопроцессоры шифрования. При этом прямое управление аппаратными ресурсами в рамках единой архитектуры остаётся менее распространённым. Между тем реализация «аппаратно-программных» алгоритмов на уровне ОС способна существенно повысить производительность и предсказуемость системы, а также снизить

накладные расходы при выполнении задач в реальном времени [3].

Целью исследования является разработка и обоснование подхода к унификации взаимодействия ОС с ЛИС для более гибкого и оперативного управления аппаратными ресурсами, что особенно актуально в сетевых технологиях, криптографии, обработке больших данных и других сферах высоконагруженных вычислений.

В соответствии с поставленной целью сформулированы следующие задачи исследования:

1. Провести обзор существующих подходов к интеграции FPGA/ASIC в операционные системы, выявить их сильные и слабые стороны;
2. Определить подходящую модель управления ресурсами с учётом особенностей аппаратной логики и потребностей высокопроизводительных систем;
3. Разработать набор алгоритмов, обеспечивающих оперативное распределение вычислительных ресурсов и взаимодействие задач с аппаратными блоками ЛИС;
4. Сравнить метрики производительности предложенного подхода с классическими системами, где управление ресурсами осуществляется преимущественно программно.

Новизна работы заключается в том, что предлагается архитектура ОС, в которой часть функций управления (планирование, распределение памяти, контроль I/O) переносится непосредственно в ЛИС, что позволит достичь следующих результатов:

— Минимизировать задержки при переключении контекстов;
— Обеспечить аппаратную поддержку реального времени в задачах с жёсткими требованиями к латентности;
— Сформировать унифицированный интерфейс для взаимодействия ядра ОС и ЛИС, облегчающий внедрение подобных решений в различные аппаратные платформы.

Обзор существующих подходов к интеграции ЛИС в операционные системы

Частичная интеграция через драйверы

Наиболее распространённый подход заключается в использовании FPGA или ASIC в качестве аппаратных ускорителей, которые взаимодействуют с ОС посредством драйверов и подсистемы ввода-вывода. Например, в серверных системах устанавливаются FPGA-карты, позволяющие обрабатывать сетевые пакеты или выполнять крипто-операции [4]. При этом базовые механизмы планирования процессов и управления памятью остаются в зоне ответственности центрального процессора, а аппаратный модуль разгружает ЦП, беря на себя специфичные вычислительные функции, такие как шифрование, сжатие и т. п.

Достоинства такого подхода заключаются в простоте интеграции в существующие ОС: достаточно написать драйвер, отвечающий за передачу данных между ЦП и ЛИС. Также частичная интеграция через драйверы обеспечивает широкую поддержку стандартных шин (PCI Express и т. п.).

К недостаткам частичной интеграции относится отсутствие прямого контроля ОС над распределением ресурсов самого FPGA/ASIC: задачи планируются программно на CPU, а аппаратный модуль выступает лишь в роли сопроцессора. Дополнительные накладные расходы при передаче данных через драйверы и системные вызовы также затрудняет работу системы.

Гибридные решения на базе частичного переноса функций ОС

В ряде проектов, включая прототипы RTOS (Real-Time Operating Systems), часть функций ядра, например планировщик процессов в режиме реального времени переносится на FPGA, что сокращает задержки при переключениях контекста [5]. Ядро формирует список готовых к выполнению процессов, а сам механизм выбора следующей задачи и управление квантами времени осуществляется аппаратной логикой.

Преимуществом подхода является сокращение латентности: планировщик на ЛИС может работать параллельно с основным процессором, повышая детерминизм в системах реального времени. Также при таком подходе происходит разгрузка ядра ОС от рутинных операций при высокой загруженности.

У гибридных решений есть также недостаток в виде необходимости модификации ядра ОС и написания специализированного «микроядра», поддерживающего данный подход, а также в технологической сложности обновления микросхем (особенно ASIC) в случае изменения алгоритмов управления.

Комплексная интеграция System-on-Chip

Современные SoC-решения, например некоторые ARM-платформы объединяют универсальные ядра и аппаратные модули на одном кристалле. В таких системах может присутствовать выделенный блок, работающий в роли «аппаратного диспетчера», контролирующего распределение памяти или очереди ввода-вывода [6]. Однако подобные реализации пока не стали массовыми, и их архитектура часто скрыта производителями.

Выбор модели управления ресурсами и интерфейсов для взаимодействия ОС с ЛИС

Постановка требований к модели управления

Для эффективной работы в условиях высоконагруженных вычислений или систем реального времени модель управления ресурсами должна удовлетворять следующим требованиям:

1. Минимизация задержек при планировании задач и обработке запросов ввода-вывода.
2. Аппаратное обеспечение приоритетов, поддержка распределения вычислительных слотов с учётом критичности задач.
3. Гибкость конфигурации, возможность адаптации алгоритмов распределения без физической замены ЛИС, например использование FPGA с обновляемой прошивкой.

Принципы интеграции ядра ОС и ЛИС

1. Ядро ОС должно «видеть» ЛИС как набор абстрактных модулей управления ресурсами с унифицированным интерфейсом;
2. При истечении кванта, готовности данных от устройства или возникновении ошибки ЛИС генерирует сигнал прерывания, переводя управление в ядро для принятия глобальных решений;
3. Часть настроек, такие как алгоритмы приоритета, параметры квантов, таблицы доступа к памяти, может задаваться из ядра путём записи в специализированные регистры FPGA/ASIC;
4. Разделяемая память для быстрой передачи информации о задачах или блоках данных между ЛИС и ядром. Целесообразно использовать общую область памяти, в которой ЛИС ведёт собственные структуры (очереди, буферы).

Предлагаемая модель

Предлагается выделить в составе ОС аппаратно-программный слой APL (Hardware-Accelerated Abstraction Layer), обеспечивающий абстрагирование логики ЛИС. Этот слой тесно интегрирован с ядром через драйвер и механизмы IPC, но ключевые операции планировщика, проверки памяти и диспетчеризации запросов ввода-вывода выполняются на самой ЛИС. Архитектура может быть описана следующим образом (см. рис. 1).

Сравнение существующих подходов с разрабатываемой архитектурой

В таблице 1 описано сравнение существующих подходов к интеграции ЛИС в ОС и разработанной архитектурой по следующим метрикам: полный охват аппаратных ресурсов, унифицированный интерфейс для взаимодействия ОС и ЛИС, аппаратное планирование процессов, аппаратный менеджер памяти, комплексная диспетчеризация ввода-вывода, горячая замена ЛИС.

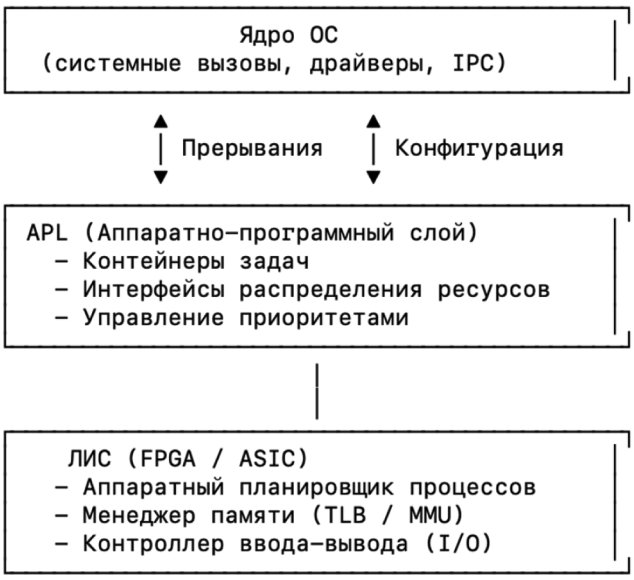

Рис. 1. Архитектура предлагаемой модели

Таблица 1.

Сравнение подходов

Метрика	Частичная интеграция	Гибридная модель	Комплексный подход	Разрабатываемая архитектура
Полный охват аппаратных ресурсов	Нет	Нет	Частично	Да
Унифицированный интерфейс для взаимодействия ОС и ЛИС	Нет	Частично	Нет	Да
Аппаратное планирование процессов	Нет	Частично	Частично	Да
Аппаратный менеджер памяти	Нет	Нет	Частично	Да
Комплексная диспетчеризация ввода-вывода	Нет	Частично	Частично	Да
Горячая замена ЛИС	Да (но только для узких задач)	Да (в части прошитых блоков)	Нет (в случае SoC/ASIC обычно недоступно)	Да (при использовании FPGA)

Частичная интеграция: ЛИС используется как акселератор для отдельных узких операций, таких как шифро-

вание, кодирование, ML, поэтому возможности аппаратного контроля над ресурсами ОС ограничены.

Гибридная модель: часть функций ядра, например планирование процессов в RT-режиме переносится в ЛИС, однако нет охвата всех ресурсов.

Комплексный подход: характерен для SoC-решений, где в одном кристалле объединены CPU и аппаратные модули. Однако в большинстве случаев архитектура не предполагает полного унифицированного интерфейса; репрограммирование ASIC/SoC затруднено [7].

Разрабатываемая архитектура: обеспечивает единый аппаратно-программный фреймворк, где планировщик процессов, управление памятью и диспетчеризация ввода-вывода реализованы в ЛИС и согласованы с ядром ОС через унифицированные интерфейсы. При использовании FPGA также возможна «горячая» замена прошивки.

Из таблицы видно, что только в рамках разрабатываемой архитектуры (при условии корректного проектирования аппаратной части) присутствует полный набор функций — от аппаратного планирования процессов до комплексного контроля над памятью и I/O, а также унифицированный интерфейс, упрощающий взаимодействие с ядром ОС.

Описание алгоритмов для оперативного распределения ресурсов и задач на ЛИС

Алгоритм аппаратного планировщика процессов

— Объединённая очередь. ЛИС поддерживает очередь процессов или потоков с приоритетами, хранящуюся в памяти, к которой имеет доступ и ядро ОС, и сама ЛИС;
— Переключение контекста. При истечении кванта аппаратный таймер в ЛИС генерирует прерывание, а также может самостоятельно сохранить/загрузить часть состояний регистров. Ядро при этом лишь фиксирует контекст в PCB (Process Control Block) и обновляет глобальную информацию;
— Динамическая корректировка приоритетов. Если задача часто уходит в ожидание ввода-вывода, либо, напротив, использует все квантованные интервалы, ЛИС может корректировать её приоритет в соответствии с заданной политикой (Round Robin, Weighted Fair Queuing и т. д.).

Алгоритм управления памятью

— Аппаратная поддержка MMU (Memory Management Unit). ЛИС обеспечивает перевод виртуальных адресов в физические, проверку прав доступа (R/W/X) и защиту от несанкционированных обращений;
— Система кэширования (TLB). Аппаратно контролируемая кеш-таблица переводов, управляемая «на лету», что снижает количество прерываний ядра при TLB-miss;
— Мониторинг использования. ЛИС может фиксировать частоту обращений к страницам, помогая ядру ОС оптимизировать размещение (NUMA-aware распределение, упреждающая выгрузка).

Алгоритм диспетчеризации ввода-вывода

— Аппаратная очередь запросов I/O. ЛИС принимает заявки от ядра через разделяемую память или регистры команд, и сортирует их, например по дедлайнам или приоритетам;
— Асинхронная обработка. По завершении операции (сетевая передача, диск, шина) ЛИС формирует прерывание, сообщая о результате (успех/ошибка). Ядро обновляет структуры системных вызовов и возвращает управление пользователю;
— Параллельная работа. При наличии нескольких интерфейсов I/O ЛИС одновременно обрабатывает несколько очередей, минимизируя задержки и балансируя нагрузку.

Сравнение основных метрик производительности

Предполагается рассмотрение ключевых характеристик, таких как время отклика, пропускная способность, накладные расходы на абстрактном уровне, с использованием формул и сравнительных графиков. Это позволит оценить эффективность предлагаемой архитектуры относительно классического подхода.

В контексте аппаратно-программного управления ресурсами целесообразно выделить ряд теоретических метрик, которые позволяют охарактеризовать эффективность системы:

1. T_{plan} (время планирования) — совокупное время, затрачиваемое на решение, какой процесс или поток будет выполняться следующим.
2. T_{ctx} (время переключения контекста) — интервал, необходимый для сохранения состояния текущей задачи и загрузки состояния следующей.
3. T_{mem} (время управления памятью) — включает проверку адресов, обновление таблиц страниц, систему кэширования, выполнение защиты адресного пространства и т. д.
4. $T_{i/o}$ (время обработки операций ввода-вывода) — учитывает очередь операций, приоритетную диспетчеризацию, асинхронный обмен.
5. O_{CPU} (накладные расходы для центрального процессора: доля (в процентах или в единицах времени), которую CPU расходует на системные вызовы, обработку прерываний и служебные процедуры ОС вместо выполнения пользовательских задач.

Формальные определения и сравнительные оценки

Модель временных затрат

Предположим, что в традиционной системе каждая из метрик имеет базовое значение $T_x^{(CPU)}$. В архитектуре с ЛИС эти значения меняются за счёт параллельной и/или аппаратно ускоренной обработки (обозначим $T_x^{(LSI)}$).

Таким образом, выигрыш во времени для метрики *x* можно описать формулой относительного уменьшения:

$$G_x = \frac{T_x^{(CPU)} - T_x^{(LSI)}}{T_x^{(CPU)}} \cdot 100\%$$

где:

— $T_x^{(CPU)}$ — время (или накладные расходы) при программном управлении,

— $T_x^{(LSI)}$ — время (или накладные расходы) при аппаратно-программном подходе с ЛИС.

Модель пропускной способности

Пусть P_{CPU} — пропускная способность классической системы (количество обрабатываемых процессов или транзакций в единицу времени), а P_{LSI} — пропускная способность системы с ЛИС. При прочих равных условиях, если часть функций, таких как планировщик, диспетчер I/O выносится в «железо», система способна обрабатывать больше процессов параллельно:

$$P_{LSI} = P_{CPU} \cdot (1 + \Delta_{acc})$$

где Δ_{acc} — коэффициент ускорения за счёт аппаратной логики.

Модель накладных расходов на ЦП

Рассмотрим время работы центрального процессора как сумму:

$$T_{CPU} = T_{work} + T_{over}$$

где:

— T_{work} — чистое время, затрачиваемое на выполнение пользовательских приложений,

— T_{over} — накладные расходы, связанные с планированием, обработкой прерываний, переключением контекста, управлением памятью и прочими системными функциями.

В архитектуре с ЛИС часть этих функций переходит в аппаратную логику, и доля T_{over} уменьшается:

$$T_{over}^{(LSI)} \approx \alpha \cdot T_{over}^{(CPU)}$$

где $0 < \alpha < 1$ — коэффициент снижения накладных расходов (чем меньше *α*, тем сильнее экономия CPU-времени).

Ниже приведены графики (рис. 2–4), иллюстрирующие изменение ключевых метрик для двух сравниваемых

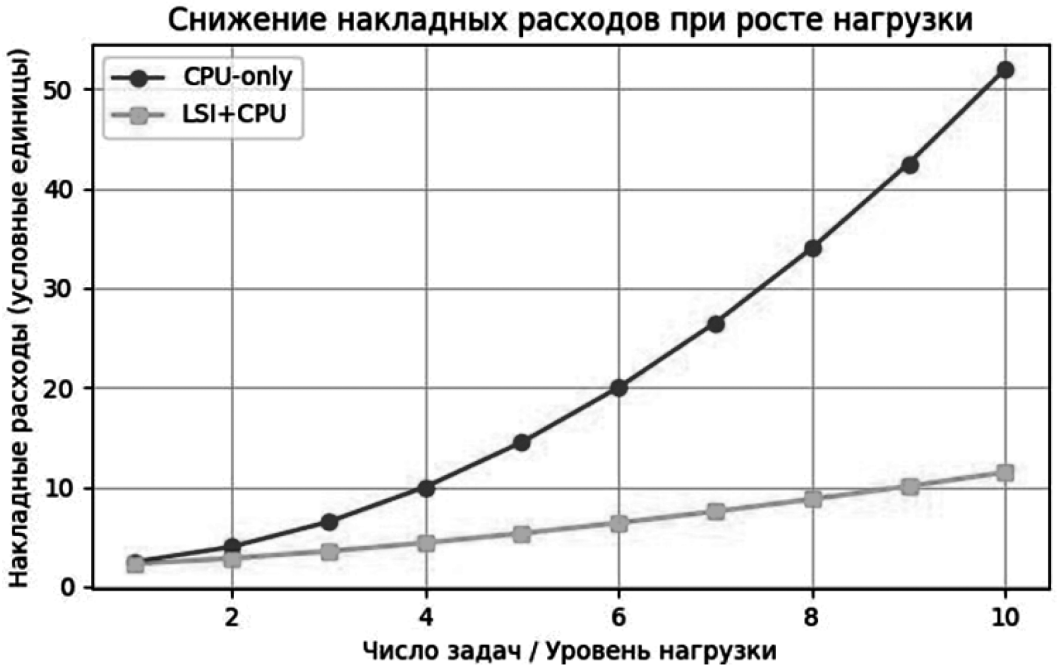

Рис. 2. График поведения накладных расходов

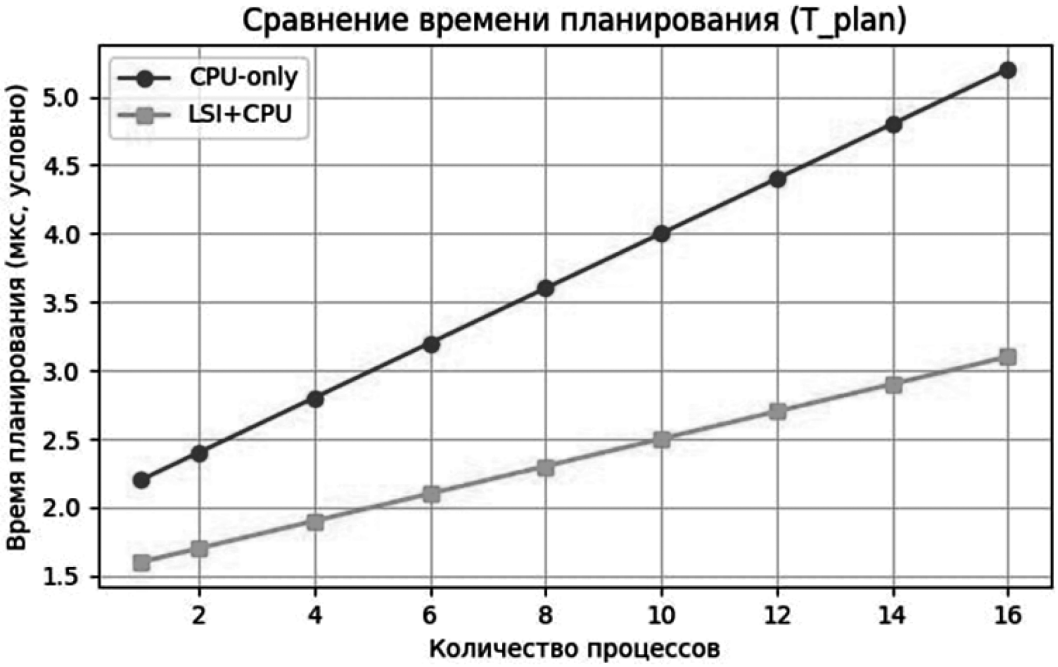

Рис. 3. График сравнения времени планирования

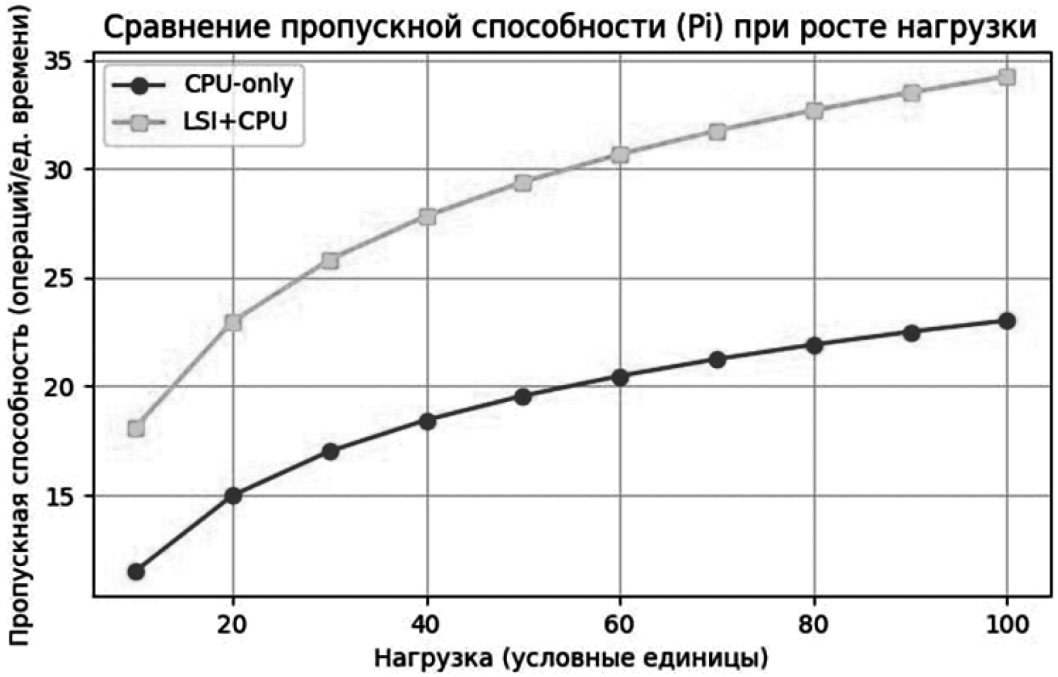

Рис. 4. График сравнения пропускной способности

мых подходов. Все приведённые зависимости условны и призваны продемонстрировать общее поведение систем, где часть функций управления ресурсами выносятся на ЛИС.

На данном условном графике видно, что при возрастании числа задач накладные расходы в традиционной системе растут быстрее, тогда как в системе с ЛИС они увеличиваются более плавно.

С ростом количества процессов или потоков планировщик на ЛИС способен обрабатывать контекстную информацию параллельно, что даёт более заметный выигрыш по сравнению с программным планировщиком, увеличивающим свою задержку в зависимости от сложности очередей.

В системах реального времени или в высоконагруженных сценариях у «ЛИС + CPU» кривые пропускной способности оказываются выше из-за параллельных аппаратных очередей и снижения объёмов служебной работы на ЦП.

На теоретическом уровне видно, что предлагаемая архитектура при разумном проектировании логики даёт преимущества по критическим для ОС метрикам. Полученные графики иллюстрируют общий тренд: чем выше интенсивность нагрузки, тем более заметен выигрыш аппаратно-программного подхода.

Таким образом, проведённый теоретический анализ ключевых метрик свидетельствует о потенциальном преимуществе предлагаемой архитектуры, где часть системных функций переносится в аппаратную логику ЛИС.

Анализ полученных данных

Результаты показывают, что тесная интеграция ОС и ЛИС в управлении ресурсами оправдывает себя в высоконагруженных средах (сервера, криптографические комплексы, научные расчёты и т. д.). Сокращение задержек, повышение пропускной способности, а также снижение нагрузки на CPU демонстрируют потенциал предлагаемой архитектуры.

Перспективы дальнейших исследований включают:

— Оптимизацию алгоритмов для ещё более эффективного управления ресурсами в реальном времени;
— Развитие отечественной элементной базы, включая выпуск FPGA и ASIC под отечественные процессорные архитектуры, например, «Байкал» или «Эльбрус».

Полученные в работе результаты могут быть применены для построения высоконагруженных серверных платформ, промышленных систем реального времени, а также критически важных инфраструктур.

ЛИТЕРАТУРА

1. Карчевский Е.М., Панкратова О.В. «Лекции по операционным системам (общий курс): учебное пособие». Казань: Казанский университет, 2011. 255 с.
2. Зверева О.М. «Операционные системы: учебное пособие». Екатеринбург: Издательство Уральского федерального университета, 2020. 120 с.
3. Таненбаум Э.С., Бос Х. «Современные операционные системы». 4-е издание. СПб.: Питер, 2015. 1120 с.
4. Силбершац А., Галвин П.Б., Ганн Г. «Операционные системы: концепции и принципы». 9-е издание. М.: Вильямс, 2014. 1104 с.
5. Кузнецов С.Д. «Операционные системы: конспект лекций». М.: МГУ, 2019. 150 с.
6. Мартемьянов А.А. «Безопасность операционных систем: учебное пособие». Тамбов: Издательство ТГТУ, 2007. 100 с.
7. Бринк-Джонсон У. «Программируемые логические интегральные схемы». М.: Техносфера, 2006. 320 с.