УДК 621.3

Научная статья

https://doi.org/10.32603/2071-8985-2023-16-5-12-23

Разработка микроконтроллерного FMU ядра на базе ПЛИС для научной космической аппаратуры

К. И. Сухачев, Д. П. Григорьев✉, Е. В. Исмагилова

Самарский национальный исследовательский университет им. акад. С. П. Королева (Самарский университет), Самара, Россия

✉grigorev.dp@ssau.ru

Аннотация. Предлагается разработанная модель FMU (Flexible Microcontroller Unit) ядра отечественной ПЛИС, цель которой состоит в получении универсального микроконтроллерного модуля для его использования в космической научной аппаратуре. FMU-ядро и соответствующая ему периферия спроектированы на ПЛИС, что позволяет сделать архитектуру всего микроконтроллера гибкой, позволяющей менять свои характеристики, без необходимости покупки других микросхем. Вся архитектура представляет собой единый конфигурационный файл, который написан на языке Verilog, объединяющий в себе блоки ядра и периферии. По желанию пользователя можно добавить либо сократить любое количество периферийных модулей. Также доступна параллельная (многоядерная) обработка данных, что недоступно на сегодняшний день в современных микроконтроллерах. Сама программа под сконфигурированный микроконтроллер написана на СИ-подобном языке, который преобразуется в двоичный пакет программы, загружаемой отдельно в память программ рассматриваемого микроконтроллера. Для преобразования исходного кода в такой пакет разработан соответствующий компилятор на языке C++. В статье приведена блок-схема архитектуры ядра, временные диаграммы различных пакетов и инструкций, приведен перечень команд ядра. Результаты тестов, также приведенные в статье, показывают малые временные затраты на выполнение некоторых инструкций по работе с внутренней и внешней ОЗУ (оперативно запоминающим и устройствами). В заключении статьи сформулированы выводы и приведены краткие характеристики ядра.

Ключевые слова: космический аппарат, микрометеороиды, микроконтроллер, ПЛИС, RISC, FMU, Verilog, компилятор

Для цитирования: Сухачев К. И., Григорьев Д. П., Исмагилова Е. В. Разработка микроконтроллерного FMU ядра на базе ПЛИС для научной космической аппаратуры // Изв. СПбГЭТУ «ЛЭТИ». 2023. Т. 16, № 5. С. 12–23. doi: 10.32603/2071-8985-2023-16-5-12-23.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Original article

Development of a Microcontroller FMU Core Based on FPGA for Scientific Space Equipment

K. I. Sukhachev, D. P. Grigoriev✉, E. V. Ismagilova

Samara National Research University (Samara University), Samara, Russia

✉grigorev.dp@ssau.ru

Abstract. A FMU (Flexible Microcontroller Unit) model of the core of a domestic FPGA is proposed with the purpose of obtaining characteristics superior in speed and type of data processing over modern domestic and

© Сухачев К. И., Григорьев Д. П., Исмагилова Е. В., 2023

foreign microcontrollers. The FMU core and its corresponding peripherals are designed on FPGA, which makes the architecture of the entire microcontroller flexible and allows its characteristics to be varied without the need to acquire other chips. The entire architecture is a single configuration file, written in the Verilog HDL and combining the core and peripheral blocks. At the request of the user, the number of peripheral modules can be either increased or reduced. Parallel (multicore) data processing is also available, which function is absent in modern microcontrollers. The program for the configured microcontroller is written in a C-style language, which is converted into a binary package of a program loaded separately into the program memory of the microcontroller in question. To convert the source code into such a package, a corresponding C++ compiler was developed. The paper provides a block diagram of the kernel architecture, time diagrams of various packages and instructions, as well as a list of kernel commands. The test results presented in the paper demonstrate small time expenditures on executing some instructions for working with internal and external RAM. Conclusions are formulated and brief characteristics of the core are given.

Keywords: spacecraft, micrometeoroids, microcontroller, FPGA, RISC, FMU, Verilog, compiler

For citation: Sukhachev K. I., Grigoriev D. P., Ismagilova E. V. Development of a Microcontroller FMU Core Based on FPGA for Scientific Space Equipment // LETI Transactions on Electrical Engineering & Computer Science. 2023. Vol. 16, no. 5. P. 12–23. doi: 10.32603/2071-8985-2023-16-5-12-23.

Conflict of interest. The authors declare no conflicts of interest.

Введение. В космической научной аппаратуре, предназначенной для регистрации микрочастиц естественного либо искусственного происхождения, необходимо иметь высокоскоростные тракты обработки сигналов [1]. Такое требование возникает ввиду высоких скоростей космических частиц естественного и искусственного происхождения. За определенное время через датчики, например ударные (пьезоэлектрические) [1] или пролетные (сеточные) [2], микрочастиц может быть целое множество, и по возможности необходимо зарегистрировать их всех. Сам математический аппарат анализа микрочастиц представляет собой несложные, но громоздкие преобразования, например двухкорреляционный способ обнаружения времени прихода механических волн на поверхности космического аппарата (в паре с преобразованием Гильберта) [3], и предсказание места удара с помощью модели нейронной сети [4]. Поэтому для повышения скорости и, как следствие, точности обработки данных необходимо использовать скоростные вычислительные модули на базе ПЛИС или микроконтроллеров.

ПЛИС – это программируемая логическая интегральная схема. Она содержит в основном логические вентили, из которых синтезируется большая логическая структура, содержащая триггеры, счетчики, регистры и прочий набор элементов, в совокупности получая требуемый пользователю механизм разрабатываемого устройства. Логическая схема синтезируется либо с помощью программного кода на языке Verilog/VHDL, либо с помощью схемотехнического редактора. ПЛИСы подразделяются на две большие группы – CPLD и FPGA. Первый тип удобно применять в случае простых коммутаций в промышленных системах, поскольку в ней содержится малое количество вентилей без каких-либо дополнительных модулей, кроме встроенной памяти с конфигурационной программой. FPGA содержит больше логических вентилей, объединенных в макроблоки. Помимо этого, у FPGA намного выше скорость и имеется аппаратная поддержка блоков цифровой обработки сигналов (DSP), но отсутствует внутренняя память, поэтому конфигурация ПЛИС при включении происходит с использованием внешней памяти. При этом ПЛИС не требует большого времени работы устройства, поскольку практически все операции внутри выполняются параллельно. Однако ПЛИС сложна с точки зрения построения программного обеспечения, например операционных систем, реализаций Ethernet-серверов и т. д. Также ПЛИС не имеет пошаговой внутрисхемной отладки с использованием исходного кода на языке Verilog.

Микроконтроллеры, в свою очередь, проще в плане программирования на языках высокого уровня (СИ/С++). Разработка программного обеспечения занимает меньше времени ввиду более понятного синтаксиса языка, архитектуры построения кода. Стоит отметить, что большинство модулей уже реализовано на сертифицированных библиотеках. Микроконтроллеры также удобнее использовать при отладке программного обеспечения с помощью как программного симулятора, так и внутрисхемного отладчика.

Микроконтроллер имеет уже готовый комплект периферийных модулей, которые запускаются в рамках нескольких строк кода. Однако микроконтроллеры недостаточно хороши по скорости обработки информации ввиду единственного вычислительного ядра. На них нельзя организовать параллельную обработку сигналов. Также размер периферии микроконтроллера (количество таймеров, линий прерывания) и ее гибкость (многофункциональность) сильно ограничены и разнятся в зависимости от поколения и модели микроконтроллера.

Настоящая статья посвящена разработке нового микроконтроллерного ядра на базе отечественной ПЛИС и внешней отечественной элементной базы, чтобы совместить достоинства ПЛИС и микроконтроллера [3].

Описание FMU-ядра. Разрабатываемое ядро называется Flexible Microcontroller Unit – FMU (гибкий микроконтроллерный блок). Такое название оно получило, поскольку достоинством микроконтроллера служит перестраиваемая под требования пользователя архитектура ядра. Сама периферия микроконтроллера также может в любой момент времени быть переконфигурирована на разный набор интерфейсных блоков, что исключает необходимость поиска и приобретения других микросхем с нужными интерфейсами. Вследствие этого в рассматриваемом микроконтроллере все целиком подвергается программированию. Ядро и периферия программируются для ПЛИС на языке Verilog, а программа для сконфигурированного ядра и периферии пишется на СИ-подобном языке. Для преобразования СИ-подобного языка в двоичный код программы микроконтроллера разработан соответствующий компилятор на языке C++, который на выходе генерирует уже готовые пакеты для программирования микроконтроллера. Архитектура ядра и набор инструкций реализованы на основе предыдущей модели IP-ядра, рассматриваемой в [5]. Схема новой архитектуры ядра показана на рис. 1.

Ядро имеет гарвардскую архитектуру памяти и содержит отдельные шину данных и шину периферии. Разрядность адреса памяти программ составляет 21 бит, разрядность шины инструкций – 16 бит, однако сами инструкции могут иметь как 16-разрядный, так и 32-разрядный формат. Структура командного слова ядра (инструкция + данные) показана на рис. 2.

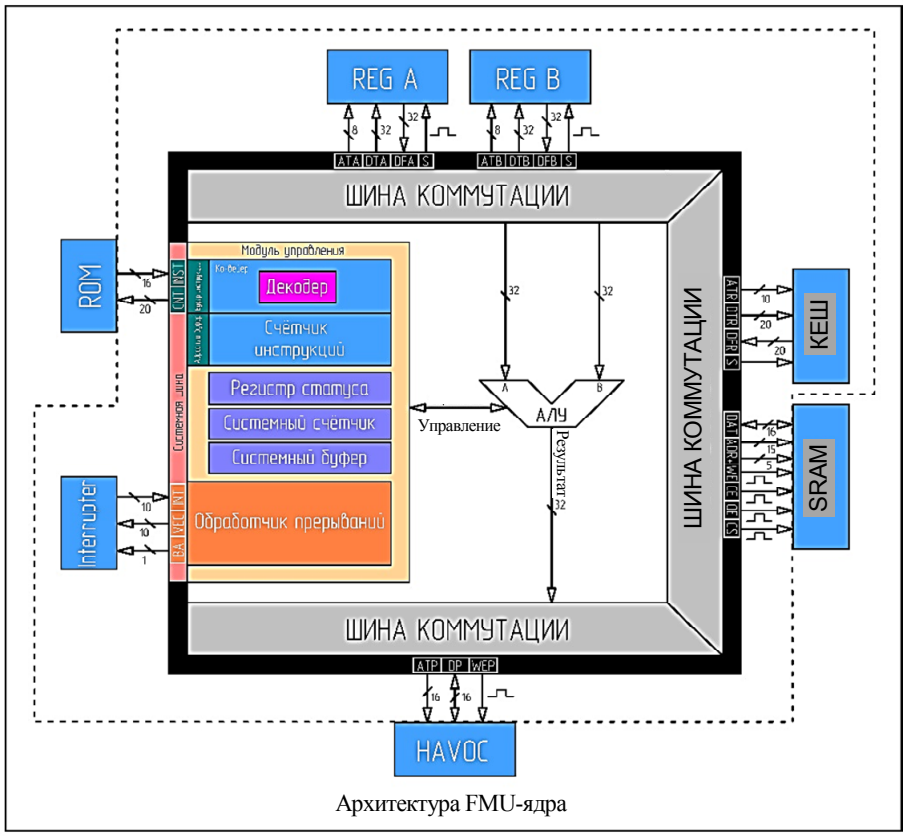

Рис. 1. Краткая схема разработанной архитектуры FMU-ядра
Fig. 1. A brief diagram of the developed FMU core architecture

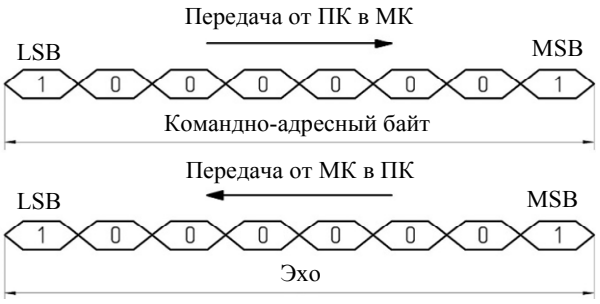

Рис. 2. Формат командного слова
Fig. 2. Diagram of command word

Формат командного слова состоит из основной и дополнительной частей слова. Одно слово – это 2 байт информации. Основная часть состоит из кода команды (выбирающей инструкцию ядра), далее следует 1-й операнд, содержащий (в зависимости от определенной инструкции) следующую информацию:

1) адрес регистра;
2) адрес младшего сектора ОЗУ;
3) адрес части периферии;
4) дополнительный указатель.

Дополнительная часть командного слова содержит информацию (в формате 1 слова):

1) информация из памяти программ (переменные/константы);
2) полный адрес ОЗУ;
3) полный адрес периферии;
4) адреса регистров блоков A и B памяти.

Ввиду отсутствия внутренней памяти в отечественных ПЛИС, ядро спроектировано под управление внешней памятью программ (ROM), в которой содержатся слова по 16 бит. Поскольку внешняя память имеет рабочую скорость ниже скорости работы ядра, в архитектуре предусмотрена поддержка выполнения команды до ее полного считывания. Весь процесс считывания и выполнения команды происходит в формате конвейера, что повышает быстродействие процессора.

Основной сегмент ядра – это АЛУ (арифметико-логическое устройство). Оно выполняет некоторые арифметические операции между двумя числами A и B, которые выставляются по 32-разрядным шинам. Блоком АЛУ управляет модуль управления ядра, который исполняет поступающие из ROM инструкции. В нем находится счетчик инструкций, буфер инструкций и декодер инструкции. В процессе работы счетчик программ взаимодействует с ROM (память программ) и в буфер инструкций загружается текущая инструкция программы. Счетчик инструкций, в свою очередь, управляет буфером инструкций, направляя команды из него в декодер [6]. Описанная последовательность представляет собой линию, из которой как раз формируется конвейер обработки инструкций. В процессе декодирования инструкции начинается выполнение этой инструкции с помощью периферии на системной/коммутационной шине или на АЛУ. После арифметической операции результат записывается в регистровую память. При необходимости АЛУ выставляет необходимые флаги статуса через линию управления АЛУ.

Флаги статуса находятся в регистре статуса ядра, представляющего собой 32-разрядный регистр. Всего флагов 4: нулевого результата; переполнения результата; смены знака результата (с положительного на отрицательный и наоборот); критической ошибки (исключения).

Остальные 28 бит отводятся для записи адреса возврата в основную программу, в момент перехода в обработчик прерывания. По этому адресу счетчик программы переключается на раздел строки ROM в том месте, откуда он перешел в требуемый обработчик. Это не касается перехода по функциям (меткам), так как в этом случае адрес хранится непосредственно в последней строке кода, что известно на стадии компиляции.

Регистровая память ядра составляет 5 блоков, из которых 3 блока располагаются внутри и 2-снаружи. Внутренние блоки REG представляют собой регистры общего назначения (РОН) в банках A и B по 1 Кбайт (256 регистров по 32 бит каждый), кеш ядра 2.5 Кбайт (1024 регистра по 20 бит каждый). Внешние блоки представляют собой внешнюю статическую ОЗУ (SRAM) до 2 Мбайт, и внешнюю 16-рично-адресуемую расширенную область HAVOC (Hexidicimal Adressable Vast Outer Connection). Последнюю можно применять для подключения модулей периферии – микросхем накопления данных, передатчиков, контроллеров различных интерфейсов, контроллеров памяти (DRAM, DDR), сопроцессорных модулей (DSP) и т. д.

Каждый банк памяти имеет свой набор шин для управления. Перечень сокращений шин управления указан в табл. 1.

Банк кеша, представляет собой 20-разрядную оперативную память ядра – младший сектор ОЗУ, 256 слов которого адаптированы под СТЕК. Ее объем составляет от 2.5 до 100 Кбайт. Благодаря большой разрядности кеш может хранить ссылки на адреса программы целиком. Также, в отличие от

Табл. 1. Расшифровка сигналов шины коммутации ядра
Tab. 1. Description of signals of FMU commutation bus

Сокращение	Расшифровка	Разрядность, бит
ATA (Address To Reg A)	Шина адреса в банк Reg A	8
DTA (Data To Reg A)	Шина данных в банк Reg A	32
DFA (Data From Reg A)	Шина данных из банка Reg A	32
S (Strob)	Стробирующий сигнал	1
DAT (Data)	Шина данных (двунаправленная) для SRAM	16
ADR+ (Address+)	Расширенная шина адреса в SRAM (15 линий + 5 линий)	15 + 5
WE (Write Enable)	Сигнал разрешения записи в SRAM	1
CE (Chip Enable)	Разрешение работы с выбранным банком SRAM	1
OE (Out Enable)	Сигнал разрешения на вывод данных из SRAM	1
CS (Chip Select)	Сигнал выбора банка SRAM	1
ATP (Address To Peripheral)	Шина адреса во внешний периферийный модуль	16
DP (Data Peripheral)	Шина данных во внешний периферийный модуль	16
WEP (Write Enable Peripheral)	Разрешение записи в периферийный внешний модуль	1
BA (Branch Archive)	Сигнал флага выполнения прерывания	1
VEC (Vector)	Шина адреса в банк Reg A	10
INT (Interrupt)	Шина адреса в банк Reg A	10
CNT (Counter)	Шина счетчика инструкций	16
INST (Instruction)	Шина инструкций от памяти программ (ROM)	20

SRAM, этот кеш во много раз быстрее и доступен во всех модулях ядра.

Статическая оперативная память SRAM подключается как внешний банк памяти, для расширения адресного пространства ОЗУ, имеющей вторую шину адреса, которая помимо стандартных 15 линий может задействовать еще 5 линий адреса. Это сделано ввиду того, что память программ поддерживает не все 20 разрядов, а только 16, поэтому данные придется записывать по очереди, т. е. в два такта (сначала 15, потом оставшиеся 5 бит), что в результате запишет 20-битное число. SRAM – это старший сектор ОЗУ. При работе программы программисту необходимо позаботиться о расширении адресного пространства, так как архитектурой ядра это не предусмотрено. На рис. 3 показана организация адресного пространства. Младшему сектору кеша выделяется ширина адресного пространства от 0 до 1023. В зависимости от ПЛИС эта ширина может изменяться, поэтому следующий сектор во внешней памяти начинается с 32768 и заканчивается 65535. Переход из встроенного кеша во внешнюю SRAM происходит выставлением старшего бита (№ 15). После

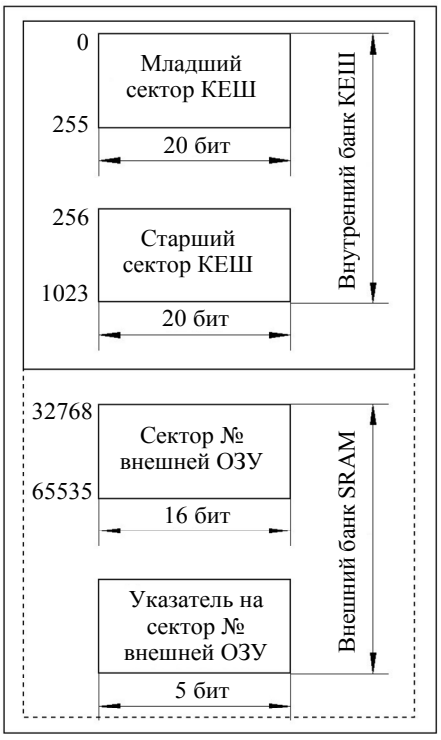

Рис. 3. Организация адресного пространства ОЗУ
Fig. 3. Organization of the RAM address space

выставления этого бита адрес автоматически становится адресом внешней SRAM. Стоит отметить, что блоки кеша и SRAM (включая расширение SRAM) обособлены адресной шиной, ввиду чего появляется возможность организовать многопоточность программы (т. е. выполнение двух подпрограмм независимо в разных банках памяти).

Также ядро поддерживает внешнее расширение HAVOC, которое позволяет подключать как микросхему памяти, так и любой другой интерфейс, обеспечивая при этом высокую скорость обмена информации и обособленность ввиду адресных шин.

Блок обработчика прерываний состоит из двух шин и одной линии флага. Флаг сообщает ядру, что обработчик прерывания выполняется. Шина VEC формирует номер (адрес) вектора прерывания для вызова соответствующего обра-

Табл. 2. Список переходных команд FMU ядра
Tab. 2. List of transitions commands of FMU core

Команда	Код инструкции	Описание
ATA<=8`d#	1	Запись 8-разрядного числа из памяти программ (ROM) в блок REG_A
ATB<=8`d#	2	Запись 8-разрядного числа из памяти программ (ROM) в блок REG_B
ATR<=16`d#	3	Запись 16-разрядного числа из памяти программ (ROM) в регистр адреса ОЗУ
ATP<=16`d#	3	Запись 16-разрядного числа из памяти программ (ROM) в регистр адреса периферии
RAM##<=A#	15	Запись в ОЗУ по адресу, полученному из текущей инструкции ##, содержимого младшей области REG_A с адресом #, также полученным из инструкции
RAM##<=B#	16	Запись в ОЗУ по адресу, полученному из текущей инструкции ##, содержимого младшей области REG_B с адресом #, также полученным из инструкции
PRH##<=A#	17	Запись в периферию по адресу, полученному из текущей инструкции ##, содержимого младшей области REG_A с адресом #, также полученным из инструкции
PRH ##<=B#	18	Запись в периферию по адресу из текущей инструкции #, данных по адресу ##, полученной из этой же инструкции
jumpB#	44	Переход на инструкцию, имеющую значение, хранящееся в REG_B по адресу #
jumpRAM#	45	Переход на инструкцию, имеющую значение, хранящееся в ОЗУ по адресу #

Табл. 3. Список арифметических команд FMU ядра
Tab. 3. List of arithmetical commands of FMU core

Команда	Код инструкции	Описание
A#<=A##*B###	57	Результат умножения операндов из REG_A и REG_B, по адресам ## и ### соответственно, записывается в РОН A по адресу #
A#<=A##/B###	58	Результат деления операндов из REG_A и REG_B, по адресам ## и ### соответственно, записывается в REG_A по адресу #
A#<=A##-B###	76	Результат разности операндов из REG_A и REG_B, по адресам ## и ### соответственно, записывается в REG_A по адресу #
A#<=A##+B###	75	Результат суммы операндов из REG_A и REG_B, по адресам ## и ### соответственно, записывается в REG_A по адресу #
A#<=A##andB###	68	Результат побитового "И" из REG_A и REG_B, по адресам ## и ### соответственно, записывается в REG_A по адресу #
A#<=A##orB###	70	Результат побитового "ИЛИ" операндов из REG_A и REG_B, по адресам ## и ### соответственно, записывается в REG_A по адресу #
A#<=A##xorB###	72	Результат побитового исключения "ИЛИ" операндов из REG_A и REG_B, по адресам ## и ### соответственно, записывается в REG_A по адресу #

Табл. 4. Список дополнительных команд FMU ядра
Tab. 4. List of advanced commands of FMU core

Команда	Код инструкции	Описание
MEM<=5`d#	111	Изменение рабочего сектора ОЗУ
int_off	120	Отключение глобальных прерываний
int_exit	127	Выход из обработчика прерываний
sleep	126	Переход в спящий режим
delayA#	125	Аппаратная задержка на значение регистра REG_A по адресу #

ботчика прерываний либо для определенных действий. Шина INT по сути – обратная связь от обработчика прерываний, чтобы сообщить ядру, какое прерывание было вызвано. Прерывание управляет логическим блоком Interrupter. Имеется поддержка 1024 векторов прерывания ввиду 10-разрядной шины ($2^{10} = 1024$).

Список инструкций FMU ядра. Перечень некоторых инструкций (мнемоника, код и описание инструкции) приведен в табл. 2–4.

Описание модуля отладки микроконтроллера на базе FMU ядра. Для первичной отладки ядра применялась импортная ПЛИС от компании «Altera». Для выбора модели микросхемы ПЛИС использовалась табл. 5, созданная в результате виртуальных (программных) тестов. Самым подходящим вариантом послужила модель EP3C16 (в дальнейшем EP3C16Q240), поскольку у нее самое малое время отклика по прерыванию и при этом малый занимаемый объем конфигурации. Целевой ПЛИС – аналогом отечественной ПЛИС, является 5578ТС034.

Сконфигурированный микроконтроллер поддерживает 14 периферийных модулей: SPI; I2C; I2S; USART; PWM; Timer/Counter; DCMI; SIINT; GPIO; MKO; Ethernet; USB; контроллеры памяти (FLASH, DRAM, DDR); контроллер RTC (Real Time Clock).

Стоит отметить, что количество периферийных блоков может варьироваться в зависимости от нужд пользователя. Структурная схема отладочного модуля FMU микроконтроллера и отладочный модуль на печатной плате показаны на рис. 4 и 5 соответственно. Модуль построен на импортной элементной базе, которая полностью заменяема отечественными аналогами.

Табл. 5. Результаты испытаний FMU ядра на разных микросхемах ПЛИС от «Altera»
Tab. 5. Results of FMU core tests on different FPGA chips from «Altera»

Параметр	ПЛИС							
	5578 ТС034	EPF10K100 EBC356-3	EPF10K100 EBC356-1	EP2C8F 256C6	EP3C16 U484C6	EP3C40 F484C6	EP3C55F 780C8	EP3C120 F780C7
Занимаемый объем, %	4181 (84 %)	4181 (84 %)	4181 (84 %)	2820 (35 %)	2681 (17 %)	2686 (7 %)	2688 (5 %)	2700 (3 %)
Занимаемая память, %	36864 (75 %)	36864 (75 %)	36864 (75 %)	36864+ (20 %)	36864+ (7 %)	36864+ (5 %)	36864+ (5 %)	36864+ (<2 %)
Кол-во РОН	512	512	512	512	512	512	512	512
КЕШ ОЗУ, Кбайт	2.5	2.5	2.5	2.5+	2.5+	2.5+	2.5+	2.5+
MIPS, операций в секунду	7.4	7.4	12.9	25.4	35.3	35.5	27.7	32.4
Максимальная частота, МГц	33.3	33.3	58.4	79	159	160	125	146
Время отклика на прерывания (не более), нс	330	330	187	132	69	70	88	75

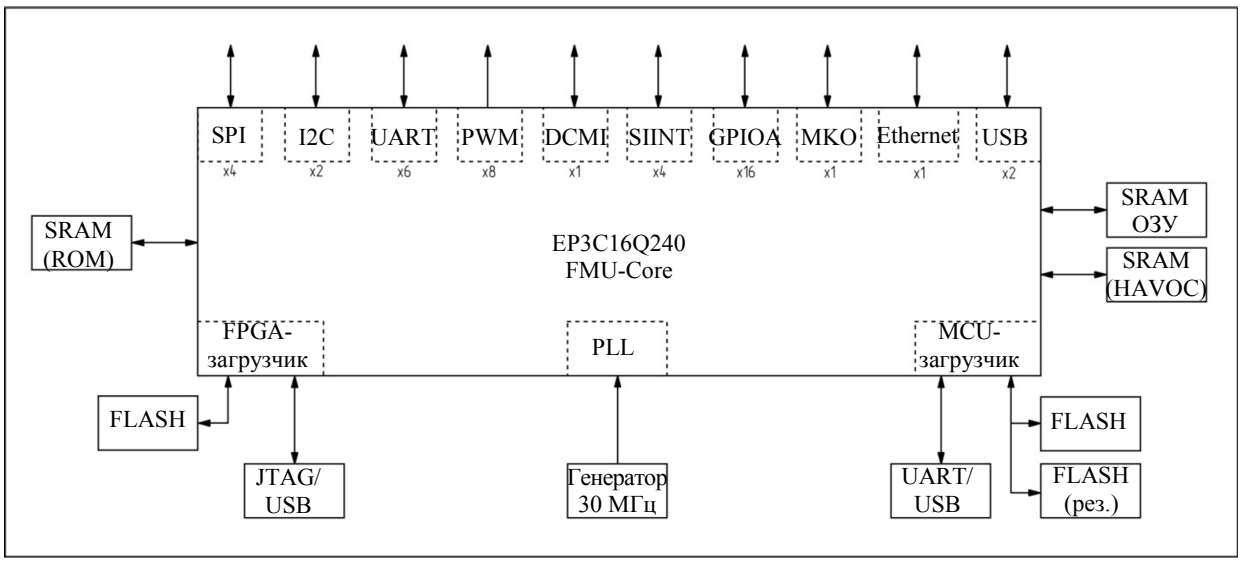

Рис. 4. Структурная схема отладочного модуля на базе FMU микроконтроллера
Fig. 4. Diagram of a debug module based on a FMU microcontroller

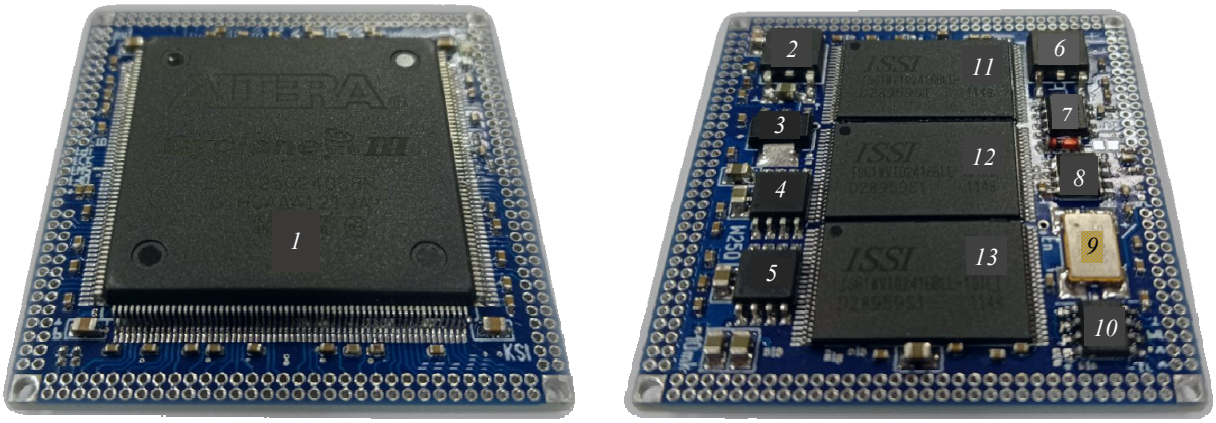

Рис. 5. Отладочный модуль FMU микроконтроллера на базе ПЛИС EP3C16:
1 – ПЛИС EP3C16; *2* – стабилизатор 3.3 В; *3* – стабилизатор 1.2 В; *4* – основная ROM; *5* – резервная ROM;
6 – стабилизатор 2.5 В; *7* – транзисторная сборка защиты от переполюсовки; *8* – FLASH для конфигурации ПЛИС;
9 – кварцевый генератор; *10* – часы реального времени; *11* – SRAM; *12* – HAVOC–SRAM; *13* – ROM–SRAM
Fig. 5. Debug module of FMU microcontroller based on FPGA EP3C16:
1 – FPGA EP3C16; *2* – 3.3 V stabilizer; *3* – 1.2 V stabilizer; *4* – main ROM; *5* – backup ROM;
6 – 2.5 V stabilizer; *7* – transistor for reverse polarity protection; *8* – FLASH for FPGA configuration;
9 – quartz oscillator; *10* – real-time clock; *11* – SRAM; *12* – HAVOC–SRAM; *13* – ROM–SRAM

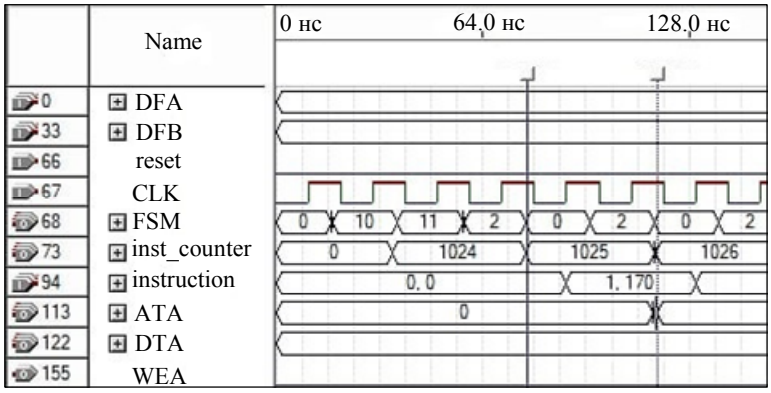

Рис. 6. Скриншот временных диаграмм сигналов ядра в момент запуска
Fig. 6. Screenshot of time diagrams of the kernel signals at the time of startup

Модуль с ПЛИС подключен к генератору на частоту 50 МГц, которая умножается блоком PLL на максимально поддерживаемую частоту работы ПЛИС. Также к модулю подключаются две микросхемы FLASH, одна из которых содержит конфигурационный файл вентилей для формирования микроконтроллера на ПЛИС, а вторая – программу для выполнения микроконтроллером. Также имеется доступ для конфигурирования ПЛИС через интерфейс JTAG/USB и загрузки/отладки программы под микроконтроллер через интерфейс UART/USB.

Результаты тестов в ПО Quartus. В момент старта ядра первые такты CLK начинаются с нулевой инструкции (instruction), как видно из рис. 6. Здесь содержится одно командное слово (0.0). Далее следует первая инструкция (1.170).

Следующие инструкции идут в зависимости от заложенной в память программы. На рис. 7 и 8 показана единая временная диаграмма (в разных временных интервалах) работы с внутренней и внешней ОЗУ соответственно.

На рис. 7 видно, что после нулевой инструкции следует инструкция записи в системный регистр адреса 50 регистра общего назначения REG A (instruction = 1.50, ATA = 50). Инструкция содержит одно слово и на ее выполнение ушло примерно 110 нс. Далее следует инструкция записи в системный регистр адреса ОЗУ (instruction = 3.1), по которому нужно записать число «1» (instruction = 0.1). Эта инструкция содержит уже два слова, и время ее выполнения – примерно 100 нс. Далее следует инструкция записи по установленному адресу ОЗУ значения из регистра общего назначения (instruction = 6. 1). Потом повторяется операция записи числа в ОЗУ, но 3-й и 4-й операнды данных содержат числа 255 и 1.

Стоит отметить, что поскольку адрес ОЗУ стал больше максимального значения адресной шины кеша (255.1 = 65280), то ядро переключилось на внешнее ОЗУ. Адрес ОЗУ складывается из младшего и старшего байтов, которые склеиваются в одно 16-разрядное адресное число. Если адрес получается больше 255, то ядро переключается на шину внешней ОЗУ. Также на рис. 8 видно, что когда постоянно выставлена одна и та же инструкция (instruction = 6.3), она будет выполняться постоянно, пока следует тактирование. В данном случае инструкция записывает данные из регистра общего назначения в ОЗУ, после чего, инкрементируя адреса обоих банков памяти (ОЗУ

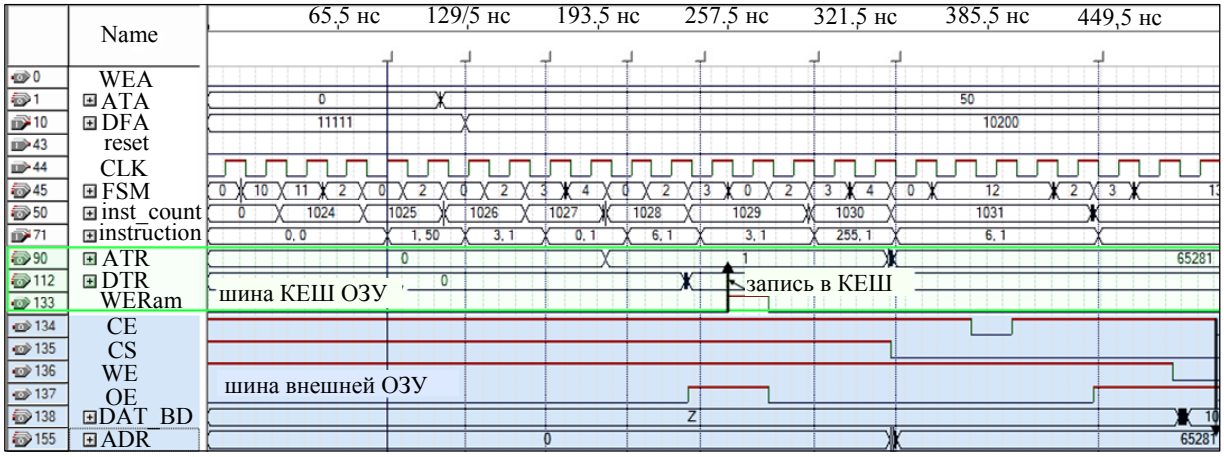

Рис. 7. Скриншот временных диаграмм работы с внутренней и внешней ОЗУ (ч. 1)
Fig. 7. Screenshot of time diagrams of working with internal and external RAM (part 1)

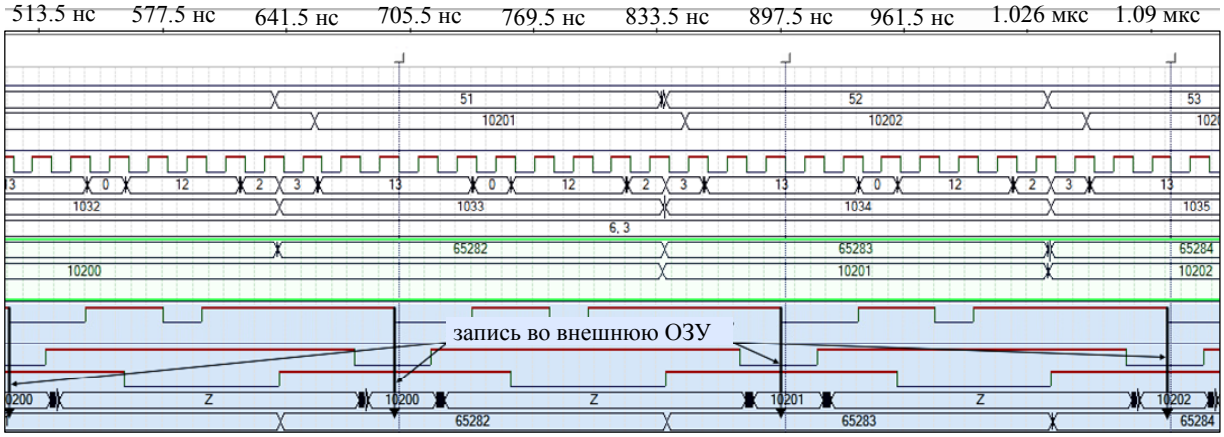

Рис. 8. Скриншот временных диаграмм работы с внутренней и внешней ОЗУ (ч. 2)
Fig. 8. Screenshot of time diagrams of working with internal and external RAM (part 2)

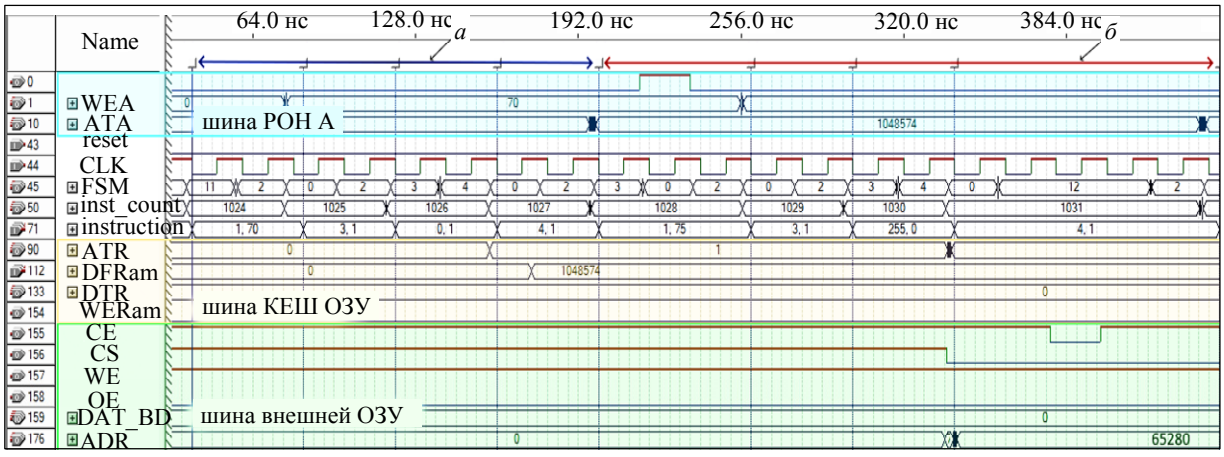

Рис. 9. Скриншот временных диаграмм работы с внутренней и внешней ОЗУ
Fig. 9. Screenshot of time diagrams of working with internal and external RAM

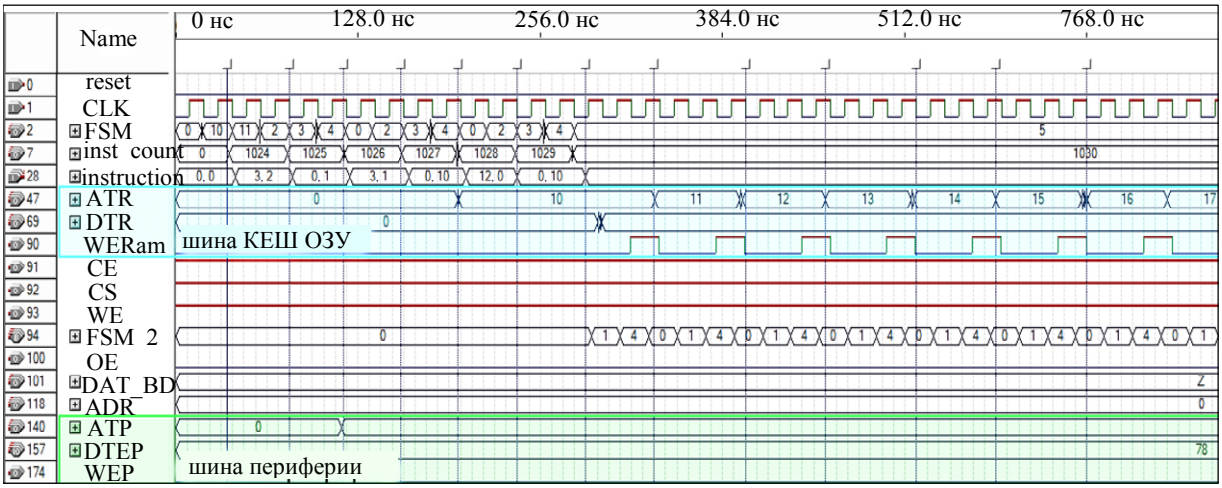

Рис. 10. Скриншот временных диаграмм записи данных из периферии в ОЗУ
Fig. 10. Screenshot of time diagrams of writing data from the periphery to RAM

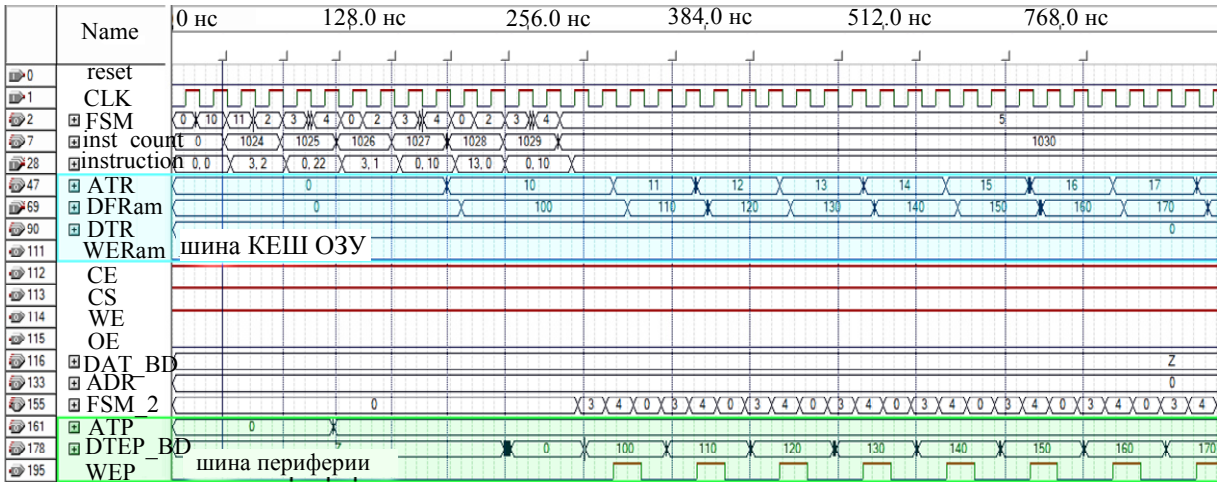

Рис. 11. Скриншот временных диаграмм записи данных из ОЗУ в периферию
Fig. 11. Screenshot of time diagrams of writing data from RAM to peripherals

и регистра общего назначения). Это прослеживается на линиях ADR внешней ОЗУ и ATA в регистре общего назначения. Поскольку в нескольких ячейках регистра общего назначения записаны числа, увеличенные на 1, то на шине данных ОЗУ DAT_BD с каждым тактом проходит эта последовательность чисел. Таким образом, в разные ячейки ОЗУ записывается возрастающая последовательность чисел из регистра общего назначения.

На рис. 9 приведен еще один процесс обмена данными между внешней и внутренней ОЗУ и регистрами общего назначения. Первая инструкция адрес регистра общего назначения (instruction = 1.70). Далее устанавливается адрес ОЗУ (instruction = 3.1; 0.1), после чего происходит запись значения из регистра общего назначения в ОЗУ (instruction = 4.1). Данные записываются в КЕШ. На следующем этапе снова устанавливается адрес регистра общего назначения, адрес ОЗУ, но данные копируются уже из ОЗУ в регистр общего назначения. На рис. 10 и 11 показан пример выполнения инструкций 12 и 13, которые передают данные между ОЗУ и периферией.

В начале, после нулевой инструкции, происходит запись адреса шины периферии (instruction = 3.2; 0.1). Далее выставляется адрес ОЗУ (instruction = 3.1; 0.10). Выставления адресов прослеживаются на линиях ATP и ATR. Затем следует команда записи данных из периферии в ОЗУ (в количестве 10 байт) (instruction = 12.0; 0.10), после чего на линии DTR появляется число, сопровождаемое стробовыми тактами WERam.

На рис. 11 повторяется начальная инициализация адресов ОЗУ и периферии, после чего следует инструкция записи данных из ОЗУ в периферию (также 10 байт) (instruction = 13.0; 0.10). В процессе выполнения команды запускается

сигнал тактирования периферии WEP, который сопутствует появлению данных на линии DTFP_BD.

Заключение. Итогом выполненной работы служит получение гибкого вычислительного FMU-ядра, которое портируется на отечественной ПЛИС и поддерживает изменяемый набор периферийных блоков для требуемой задачи. Микроконтроллеры такого вида можно применять в космической научной аппаратуре, поскольку они имеют полноценный функционал современных на нынешний день микроконтроллеров, а также высокую скорость работы за счет параллельной обработки данных и дополнительных сопроцессорных блоков. Микроконтроллеры поддерживаются различными внешними микросхемами памяти, включая динамическую память типа DDR, что позволяет поднять скорость и производительность работы.

Основные характеристики ядра:

1) аппаратная поддержка операций умножения и деления;

2) поддержка 16- и 32-разрядных инструкций;

3) наличие 20-разрядной шины адреса памяти программ;

4) наличие контроллеров внешней памяти;

5) наличие 1024 векторов прерываний;

6) поддержка инструкций для потоковых операций чтения и записи из ОЗУ в ПРФ и наоборот.

Список литературы

1. Григорьев Д. П. Система контроля поверхности космического аппарата // Междунар. молодежная науч. конф. «XVI Королевские чтения», посв. 60-летию полета в космос Ю. А. Гагарина. Самара, 2021. С. 355–357.

2. Воронов К. Е., Телегин А. М., Рязанов Д. М. Концепция прибора на основе сеточной конструкции для измерения параметров микрометеороидов // Прикладная физика. 2021. № 4. С. 73–80. doi: 10.51368/1996-0948-2021-4-73-80.

3. Воронов К. Е., Григорьев Д. П., Телегин А. М. Исследование алгоритмов для системы контроля поверхности космического аппарата на основе пьезодатчиков // Авиакосмическое приборостроение. 2021. № 1. С. 40–50. doi: 10.25791/aviakosmos.1.2021.1200.

4. Воронов К. Е., Григорьев Д. П., Телегин А. М. Применение нейронной сети прямого распространения для локализации места удара микрочастиц о поверхность космического аппарата // Тр. МАИ. 2021. № 118. С. 1–35. doi: 10.34759/trd-2021-118-10.

5. Разработка высокопроизводительной вычислительной системы на базе IP-ядра для космической научной аппаратуры / К. И. Сухачев, К. Е. Воронов, А. С. Дорофеев, Д. А. Шестаков, А. А. Артюшин // Научное приборостроение. 2022. Т. 32, № 4. С. 88–106.

6. Практический курс микропроцессорной техники на базе процессорных ядер ARM-Cortex-M3/M4/M4F. Архитектура, система команд, разработка и отладка программного обеспечения на Ассемблере в интегрированной среде Keil uVision / В. Ф. Козаченко, Д. И. Алямкин, А. С. Анучин, А. А. Жарков, М. М. Лашкевич, Д. И. Савкин, Д. М. Шпак. М.: Изд-во МЭИ, 2019. 543 с.

Информация об авторах

Сухачев Кирилл Игоревич – канд. техн. наук, доцент, ст. науч. сотр. ИКП-214 (Институт космического приборостроения), Самарский университет, Московское шоссе, д. 34, к. 3, Самара, 443086, Россия.
E-mail: sukhachev.ki@ssau.ru
https://orcid.org/0000-0003-4076-5916

Григорьев Данил Павлович – инженер-конструктор ИКП-214, аспирант Самарского университета, Московское шоссе, д. 34, к. 3, Самара, 443086, Россия.
E-mail: grigorev.dp@ssau.ru

Исмагилова Елена Владимировна – лаборант ИКП-214, аспирант Самарского университета, Московское шоссе, д. 34, к. 3, Самара, 443086, Россия.
E-mail: ismagilova.ev@ssau.ru

References

1. Grigor'ev D. P. Sistema kontrolja poverhnosti kosmicheskogo apparata // Mezhdunar. molodyozhnaja nauch. konf. «XVI Koroljovskie chtenija», posv. 60-letiju poljota v kosmos Yu. A. Gagarina. Samara, 2021. S. 355–357. (In Russ.).

2. Voronov K. E., Telegin A. M., Ryazanov D. M. Koncepciya pribora na osnove setochnoj konstrukcii dlja izmerenija parametrov mikrometeoroidov // Prikladnaja fizika. 2021. № 4. S. 73–80. doi: 10.51368/1996-0948-2021-4-73-80. (In Russ.).

3. Voronov K. E., Grigor'ev D. P., Telegin A. M. Issledovanie algoritmov dlja sistemy kontrolja poverhnosti kosmicheskogo apparata na osnove p'ezodatchikov // Aviakosmicheskoe priborostroenie. 2021. № 1. S. 40–50. doi: 10.25791/aviakosmos.1.2021.1200. (In Russ.).

4. Voronov K. E., Grigor'ev D. P., Telegin A. M. Primenenie nejronnoj seti prjamogo rasprostranenija dlja lokalizacii mesta udara mikrochastic o poverhnost' kosmicheskogo apparata // Tr. MAI. 2021. № 118. S. 1–35. doi: 10.34759/trd-2021-118-10. (In Russ.).

5. Razrabotka vysokoproizvoditel'noj vychislitel'noj sistemy na baze IP-jadra dlja kosmicheskoj nauchnoj apparatury / K. I. Suhachyov, K. E. Voronov, A. S. Dorofeev, D. A. SHestakov, A. A. Artyushin // Nauchnoe priborostroenie. 2022. Vol. 32, № 4. S. 88–106. (In Russ.).

6. Prakticheskij kurs mikroprocessornoj tekhniki na baze processornyh jader ARM-Cortex-M3/M4/M4F. Arhitektura, sistema komand, razrabotka i otladka programmnogo obespechenija na Assemblere v integrirovannoj srede Keil uVision / V. F. Kozachenko, D. I. Aljamkin, A. S. Anuchin, A. A. ZHarkov, M. M. Lashkevich, D. I. Savkin, D. M. SHpak. M.: Izd-vo MJEI, 2019. 543 s. (In Russ.).

Information about the authors

Kirill I. Sukhachev – Cand. Sci. (Eng.), Assistant Professor, Senior Researcher IKP-214, Samara University, Moskovskoe shosse st., 34, k. 3, Samara, 443086, Russia.
E-mail: sukhachev.ki@ssau.ru
https://orcid.org/0000-0003-4076-5916

Danil P. Grigoriev – Design engineer IKP-214, graduate student Samara University, Moskovskoe shosse st., 34, k. 3, Samara, 443086, Russia.
E-mail: grigorev.dp@ssau.ru

Elena V. Ismagilova – Laboratory assistant IKP-214, graduate student Samara University, Moskovskoe shosse st., 34, k. 3, Samara, 443086, Russia.
E-mail: ismagilova.ev@ssau.ru

Статья поступила в редакцию 22.02.2023; принята к публикации после рецензирования 21.03.2023; опубликована онлайн 25.05.2023.

Submitted 22.02.2023; accepted 21.03.2023; published online 25.05.2023.