РАКЕТНО-КОСМИЧЕСКОЕ ПРИБОРОСТРОЕНИЕ И ИНФОРМАЦИОННЫЕ СИСТЕМЫ
2021, том 8, выпуск 1, с. 24–38

РАДИОТЕХНИКА И КОСМИЧЕСКАЯ СВЯЗЬ

УДК 629.78 DOI 10.30894/issn2409-0239.2021.8.1.24.38

Разработка бортового модуля управления на базе вычислительного IP-ядра

К. Е. Воронов, *к. т. н., доцент, voronov.ke@ssau.ru*
Самарский национальный исследовательский университет им. С. П. Королева, г. Самара, Российская Федерация

К. И. Сухачев, *к. т. н., доцент, kir.sukhachev@gmail.com*
Самарский национальный исследовательский университет им. С. П. Королева, г. Самара, Российская Федерация

Д. С. Воробьев, *vorobev.ds@ssau.ru*
Самарский национальный исследовательский университет им. С. П. Королева, г. Самара, Российская Федерация

Аннотация. В статье представлены результат реализации синтезируемого микроконтроллера в интегральных микросхемах ПЛИС небольшого объема и вариант построения системы управления бортового модуля управления на базе разработанного решения. Показана возможность создания полноценного микроконтроллера на базе ПЛИС типа 5578ТС034 и более емких микроконтроллеров. Приведено описание структуры микроконтроллера, процессорного ядра и периферии. Представлена система команд процессора. Разработаны IP-модули периферийных устройств и некоторых интерфейсов.

Предложен вариант создания системы управления с использованием разработанного микроконтроллера. В дальнейшем планируется увеличение функциональных возможностей синтезируемого микроконтроллера посредством оптимизации IP-модулей и добавления новых. При разработке системы управления применялась отечественная компонентная база.

Ключевые слова: ПЛИС, IP-ядро, микроконтроллеры

Development of Control Module Based on a Computing IP-Core

K. E. Voronov, *Cand. Sci. (Engineering), voronov.ke@ssau.ru*
Samara National Research University, Samara, Russian Federation

K. I. Sukhachev, *Cand. Sci. (Engineering), kir.sukhachev@gmail.com*
Samara National Research University, Samara, Russian Federation

D. S. Vorobev, *vorobev.ds@ssau.ru*
Samara National Research University, Samara, Russian Federation

Abstract. The article presents the result of the implementation of a synthesized microcontroller in integrated circuits of small FPGAs and a variant of building a control system for an onboard control module based on the developed solution. The possibility of creating a full-fledged microcontroller based on a type 5578TC034 FPGA and more capacious microcontrollers is shown. The description of the structure of the microcontroller, processor core and periphery is given. The processor instruction system is presented. Ip-modules of peripheral devices and some interfaces have been developed.

A variant of creating a control system using the developed microcontroller is proposed. In the future, it is planned to increase the functionality of the synthesized microcontroller by optimizing ip-modules and adding new ones. When developing the control system, a domestic component base was used.

Keywords: FPGA, ip-core, microcontrollers

Введение

В ряде случаев при разработке электронной аппаратуры приоритетным является использование отечественной элементной базы. В условиях активно расширяющейся номенклатуры возникает необходимость реализации универсального решения с возможностью применения в различных проектах.

Часто при разработке возникает ситуация, когда от системы управления (СУ) не требуется высоких скоростей обработки информации и большой вычислительной мощности, но необходимы высокая надежность и многоуровневый характер функционирования. К типовым задачам, решаемым СУ, относятся исполнение несложных алгоритмов, обработка информации с датчиков и аналого-цифровых преобразователей, сохранение обработанной информации в банке памяти, формирование пакетов информации, ведение записи событий. Иногда возникает необходимость реализации специальных функций.

Рассматривая приведенный перечень задач, решаемых системой управления, можно сделать вывод, что во многих случаях достаточно использование несложных микроконтроллеров (МК). Образцов отечественного производства с подходящими параметрами и доступных на рынке на данный момент не так уж много, например можно выделить контроллер 1986ВЕ8Т производства АО «ПКК Миландр». Данный 32-разрядный контроллер разработан на базе процессорного ядра ARM Cortex-M4F, обладает высокой производительностью и развитой периферией. Однако для решения большинства перечисленных задач СУ его возможности избыточны и с экономической точки зрения покупка иногда бывает нецелесообразной.

Другим возможным вариантом компонентной базы для реализации СУ являются интегральные микросхемы (ИМС) ПЛИС [1, 2], например производства АО «ВЗПП-С». Среди подходящих ПЛИС на данный момент можно выделить два варианта — это 5576ХС6Т и 5578ТС034 на 2880 и 4992 эквивалентных логических элемента соответственно [3]. Выбор данных ПЛИС является экономически более выгодными решениями, чем использование мощного микроконтроллера. Комбинирование функциональных модулей и процессорного ядра позволяет снизить нагрузку на процессор, тем самым его можно упростить, при этом не усложняя логику работы самих IP-модулей. Имея набор наиболее часто применяемых модулей и процессорное ядро, можно собирать аналог МК. Данный МК будет обладать уникальным, необходимым только в разрабатываемом проекте функционалом.

Сложностью на пути реализации описанной концепции является то, что доступные ПЛИС обладают небольшой логической емкостью и многие уже готовые решения, IP-ядра и модули плохо или вообще несовместимы с ними. ПЛИС серии 5576 и 5578 — аналоги семейства FLEX 10KE Intel (Altera) [3]. Особенности внутренней организации серии FLEX реализация в них какой-либо логической функции занимает в среднем на 50 % больше ресурсов ИМС по сравнению, например, с серией Cyclone IV, а задержки прохождения сигнала, как правило, выше.

Задачи

Исходя из вышесказанного были поставлены следующие задачи:

1) разработать архитектуру процессорного IP-ядра с учетом особенностей применяемой элементной базы:

– малый объем эквивалентных логических элементов ПЛИС (4992 — для 5578ТС034);

– значительные задержки прохождения сигналов;

– отсутствие аппаратных умножителей;

– малый объем доступной внутренней памяти ПЛИС (40 960–48 000 бит);

– для однократно программируемой ПЛИС — невозможности задавать начальное состояние триггеров и элементов памяти при загрузке.

В качестве внешней однократной памяти для хранения программ желательно ориентироваться на ИМС 1645РТ2У (32Кх8);

2) разработать минимальный набор команд для IP-ядра процессора, позволяющий в ручном режиме осуществлять набор ассемблерных программ;

3) разработать набор периферийных IP-модулей, необходимых для работы процессорного ядра, таких как контроллер прерываний, аппаратные таймеры-счетчики, контроллер периферии, контроллер

внешней памяти и памяти программ, IP-модули основных распространенных интерфейсов — UART, I2C, SPI;

4) разработать скоростной интерфейс без применения дополнительных ИМС-преобразователей для гальванической развязки, последовательный скоростной интерфейс для обмена между ПЛИС в границах одной платы, управляемый ШИМ-контроллер с расширенными возможностями настройки (парафазный режим, регулировка разрядности ШИМ и установка мертвого времени). Все перечисленные IP-ядра необходимо оптимизировать под используемую элементную базу;

5) на базе разработанных решений сконфигурировать синтезируемый микроконтроллер.

В данной статье приведены этапы и результаты разработки синтезируемого микроконтроллера, оптимизированного для реализации в однократно программируемых ПЛИС серии 5576 и 5578.

Описание IP-ядра процессора

Разработанное ip-ядро процессора имеет гарвардскую архитектуру. Разрядность входов арифметико-логического устройства (АЛУ) составляет 16 бит, а выход АЛУ имеет ширину 32 бита. Входы АЛУ буферизированы регистрами общего назначения (РОН): 32-битным регистром A и 16-битным регистром B. Регистр A через мультиплексор соединен с выходом АЛУ, это сделано для возможности проведения операций сложения и умножения 16-ти битных чисел. Например, операция $A = A * B$ происходит за один процессорный такт, и после умножения результат помещается в регистр A, исходное же значение регистра A стирается. Флаги переполнения в таком случае не нужны так как результат единичной операции не может превышать разрядность регистра A. Кроме РОН в ip-ядре реализованы регистры специального назначения (РСН), для работы с шинами периферии и оперативной памяти (ОЗУ), данные шины одинаковы по составу и имеют 16 бит адреса, 16 бит данных на запись и чтение, а также сигнал разрешения записи. Максимальный объем ОЗУ, который можно подключить к такой шине составляет 131 килобайт. Для ОЗУ ядра процессора используется внутренняя память ПЛИС. Для ИМС 5578ТС034 доступно всего 48 Кбит, а для 5578ТС094 уже 594 Кбит. В зависимости от выполняемой программы конфигурацию ip-ядра, набор команд, разрядность регистров, а также объем подключаемой ОЗУ можно изменять. Целесообразно часть встроенной памяти ПЛИС отдать различным исполнительным модулям, а оставшуюся память использовать в качестве ОЗУ процессора.

С точки зрения описания ЦП представляет собой единый модуль, написанный на языке Verilog, который включает в себя систему управления процессором с дешифратором инструкций, контроллером прерываний, счетчиком инструкций и контроллером шин периферии и ОЗУ, а также АЛУ, РОН, РСН и устройство маршрутизации. Структура ip-ядра процессора представлена на рис. 1.

Командное слово процессора состоит из 6 бит инструкций и 16 бит данных операнда адреса или расширения в зависимости от инструкции. Предусмотрена возможность работы с внешней 8ми битной памятью, для хранения программ, в таком случае командное слово расширяется до 24 бит (3-х байт).

Структура командного слова приведена на рис. 2. Два добавленных бита используются как указатель длины команды.

Система управления (СУ), обеспечивает координацию работы всех внутренних блоков процессора, осуществляет обработку прерываний, декодирование полученных инструкций, и, в зависимости от инструкции управляет коммутатором, соединяя нужным образом внутренние регистры, шин, входов и выходов АЛУ. СУ также производит вычисление адреса следующей команды. Один процессорный цикл системы управления состоит из последовательности следующих операций:

1) Осуществляется проверка блокировки прерываний, если такой блокировки нет, то осуществляется проверка на нахождение в прерывании. В случае, если процессор обрабатывает инструкции прерывания, то процесс продолжается в обычном режиме, т. е. осуществляется переход к следующей операции цикла. Предусмотрена возможность при выходе из порывания проверить, не возникало ли новых запросов на прерывание от периферии в момент обработки текущего.

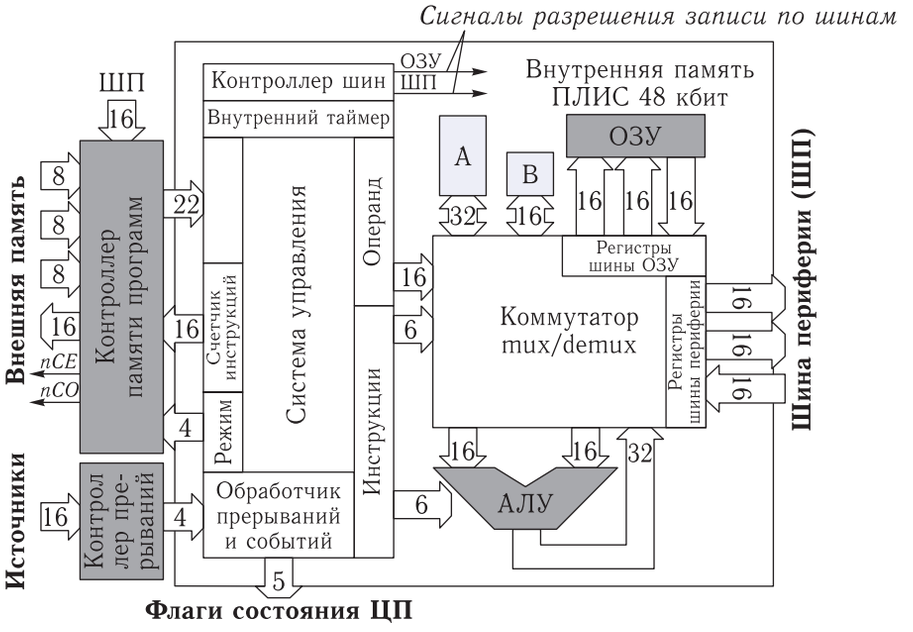

Рис. 1. Структура ip-ядра процессора

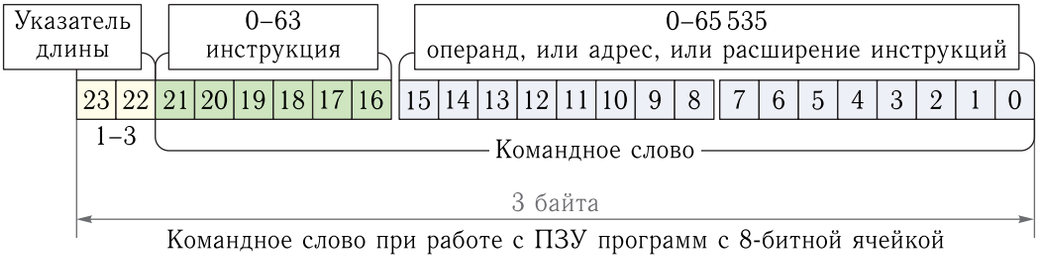

Рис. 2. Структура командного слова процессора

Данная возможность реализована с помощью памяти запросов во внешнем контроллере прерываний. На данном шаге выполняется генерация сигнала для сторожевого таймера;

2) Проверка на запрос по прерыванию. Если такой запрос детектирован, то вызывается процедура сохранения текущего значения счетчика команд и регистров в буфер, после чего происходит присвоение счетчику команд значения, соответствующего вектору детектированного прерывания. При завершении программы прерывания, происходит автоматическое восстановление из буфера значений счетчика команд и регистров, таким образом продолжается выполнение основной программы. Если запроса на прерывание нет, то система управления переходит к следующей операции.

3) Выполняется проверка на нулевое значение счетчика команд. Данное состояние может быть либо сразу после включения системы, либо после перезагрузки. В таком случае осуществляется перевод на адрес первой команды в памяти программ, данный адрес находится после области адресов переходов по прерываниям;

4) На данной стадии происходит ожидание установки значения на входе контроллера памяти программ (длительность зависит от режима работы и непосредственно от скорости ИМС внешней памяти);

5) Осуществляется декодирование инструкции, определяется ее тип, выделяется операнд и адрес, содержащиеся в командном слове;

6) На последней операции происходит исполнение текущей инструкции. Данный процесс зависит

от типа инструкции и может занимать от одного до нескольких тактов. Если выполняемая инструкция является одной из инструкций перехода, то осуществляется изменение значения счетчика команд на значение, соответствующее операнду данной инструкции или по механизму косвенной адресации, в противоположенном случае после выполнения действий, вызванных текущей инструкцией, осуществляется переход на адрес следующей команды. Адрес следующей инструкции зависит от режима работы с внешней памятью. Далее происходит возврат к первому пункту процессорного цикла.

Максимально полно о принципе и структуре процессора можно судить по его системе команд, которая представлена в табл. 1–4.

Некоторые инструкции в табл. 1 являются избыточными и служат для сокращения объема программы. В целях оптимизации можно удалить часть логики ЦП, реализующую неиспользуемые в актуальной программе инструкции. В табл. 2 представлен список команд переходов.

В список арифметических команд (табл. 3) включены операции умножения и деления на аппаратно-реализованных блоках, подключать данные модули без крайней необходимости нецелесообразно из-за отсутствия умножителей в ПЛИС5578ТС034.

Контроллер в базовой версии имеет 63 команды; с учетом расширений, доступных для некоторых из них, общее число доступных операций составляет 78. В систему управления ЦП входит также контроллер шин для с шинами ОЗУ и периферии. Его задача формировать сигналы разрешения записи по мере готовности данных на соответствующих регистрах. адреса и данных. Контроллер шин вносит задержку в один такт на выполнение операции с ОЗУ и периферией незначительно снижая общую производительность системы, однако в совокупности с полностью синхронным дизайном всей логики как самого ip-ядра, так и вспомогательных и периферийных модулей, делает систему более стабильной и непривязанной к аппаратным особенностям ПЛИС. Для верификации результатов разработки были произведены успешные сборки проекта и моделирования для разных семейств ПЛИС, таких как FLEX, MAXII, cycloneIII и cycloneIV. Для CPLD в проект пришлось вводить изменения, связанные с отсутствием внутренней оперативной памяти.

Встроенный в ip-ядро таймер управляется командой delayA, описанной в табл. 4. Особенностью данного таймера является то, что при выполнении инструкции delayA не пропадает чувствительность процессора к запросам на прерывание, и при обработке прерывания счетчик не останавливается, что дает возможность предсказуемо и точно реализовывать необходимую длительность задержки. Система обработки прерываний разделена на две части: встроенный в ядро ЦП обработчик, и внешний контроллер прерываний, являющийся приоритетным среди устройств, работающих по периферийной шине. Обработчик прерываний реализует переход к заданной области памяти программ в зависимости от вектора прерывания, а также обеспечивает буферизацию актуальных данных, содержащихся в регистрах ip-ядра. Одновременно обработчик может работать только с одним прерыванием и имеет всего 16 векторов возможных переходов, однако его возможности существенно расширяет внешний контроллер прерываний. Так, например, все модули UART вызывают прерывание с номером 4, однако, сделав запрос до контроллера прерываний по периферийной шине, можно выяснить, какой именно модуль вызвал прерывание и по какой причине. Контроллер прерываний имеет внутреннюю память, в которой хранится история последних запросов. Данная особенность позволяет при выходе из прерывания, по шипе периферии обратится к памяти контроллера прерываний и проверить историю запросов, которые могли возникнуть при обработке текущего прерывания. Если возникает ситуация с одновременным появлением запросов на прерывание, то они обрабатываются согласно заранее установленным приоритетам. При запрете обработки прерываний процессорный цикл сокращается, и основная программа выполняются быстрее. Важно, что внешний контроллер прерываний продолжает работать в полнофункциональном режиме с сохранением истории прерываний, что позволяет реализовать режим, при котором основная программа выполняется максимально быстро и программным методом периодически проверяет память контроллера прерываний.

Контроллер памяти программ (КПП) не является обязательным в том случае, если для хранения

Таблица 1. Общая система команд чтения — записи разработанного процессора

№	Инструкции	Описание
1	AX	Запись в младшие 16 бит 32-разрядного регистра A данных из текущей инструкции: **A[15:0]<=Operand[15:0]**
2	BX	Запись в 16-битный регистр B данных из текущей инструкции: **B[15:0]<=Operand[15:0]**
3	RX	Запись в 16-битный регистр адреса ОЗУ R данных из текущей инструкции: **R[15:0]<=Operand[15:0]**
4	PX	Запись в 16-битный регистр адреса периферии P данных из текущей инструкции: **P[15:0]<=Operand[15:0]**
5	RamXR	Запись данных из текущей инструкции в ОЗУ по адресу, записанному в регистре R: **RAM(R)<=Operand[15:0]**
6	RamXR+	Запись данных из текущей инструкции в ОЗУ по адресу, записанному в регистре адреса R, с его последующим инкрементированием: **RAM(R)<=Operand[15:0]; R+1**
7	PrhXP	Запись данных из текущей инструкции в периферию по адресу, записанному в регистр адреса периферии P: **PRH(P)<=Operand[15:0]**
8	PrhXP+	Запись данных из текущей инструкции в периферию по адресу, записанному в регистр адреса P, с его последующим инкрементированием: **PRH(P)<=Instruction[15:0]; P+1**
9	RamA	Запись в ОЗУ содержимого младших разрядов регистра A. При нулевом операнде после инструкции запись осуществляется по адресу из регистра R. При ненулевом операнде запись происходит в ячейку ОЗУ с адресом, равным операнду из текущей команды: **0:RAM(R)<=A[15:0]; !0:RAM(X)<=A[15:0]; X<=Operand[15:0]**
10	RamB	Запись в ОЗУ содержимого регистра B. При нулевом операнде после инструкции запись осуществляется по адресу из регистра R. При ненулевом операнде запись происходит в ячейку ОЗУ с адресом, равным операнду из текущей команды: **0:RAM(R)<=B[15:0]; !0:RAM(X)<=B[15:0]; X<=Operand[15:0]**
11	RamAB	Запись в ОЗУ содержимого младших разрядов регистра A по адресу из регистра B: **RAM(B)<=A[15:0]**
12	RamBA	Запись в ОЗУ содержимого регистра B по адресу из младших разрядов регистра A: **RAM(A)<=B[15:0]**
13	Read Prh	При расширении инструкций, равном 0, осуществляется запись из периферийного устройства с адресом из регистра P в ячейку ОЗУ с адресом из регистра R. При расширении инструкций, равном 1, осуществляется запись из периферийного устройства с адресом из регистра P в младшие разряды регистра A. При расширении инструкций, равном 2, осуществляется запись из периферийного устройства с адресом из регистра P в регистр B: **0:RAM(R)<=PRH(P); 1:A<=PRH(P); 2:B<=PRH(P)**

Таблица 1. *Окончание*

№	Инструкции	Описание
14	Write Prh	При расширении инструкций, равном 0, осуществляется запись в периферийное устройство с адресом из регистра P содержимого младших разрядов регистра A. При расширении инструкций, равном 1, осуществляется запись в периферийное устройство с адресом из регистра P содержимого регистра B. При расширении инструкций, равном 2, осуществляется запись в периферийное устройство с адресом из регистра P в содержимые ячейки ОЗУ с адресом из регистра R: **0:PRH(P)<=A[15:0]; 1:PRH(P)<=B[15:0]; 2:PRH(P)<=RAM(R)**
15	PrhBX	Запись в периферийное устройство с адресом, равным операнду текущей команды содержимого регистра B: **PRH(X)<=B[15:0]; X<=Operand[15:0]**
16	ARamX	Запись в младшие разряды регистра A содержимого ячейки ОЗУ с адресом, равным операнду текущей команды: **A[15:0]<=RAM(X); X<=Operand[15:0]**
17	BRamX	Запись в регистр B содержимого ячейки ОЗУ с адресом, равным операнду текущей команды: **B[15:0]<=RAM(X); X<=Operand[15:0]**
18	Read Ram	При расширении инструкций, равном 0, осуществляется запись в младшие разряды регистра A содержимого ячейки ОЗУ с адресом из регистра R. При расширении инструкций, равном 1, осуществляется запись в регистр B содержимого ячейки ОЗУ с адресом из регистра R: **0:A[15:0]<=RAM(R); 1:B[15:0]<=RAM(R)**
19	Read Ram+	При расширении инструкций, равном 0, осуществляется запись в младшие разряды регистра A содержимого ячейки ОЗУ с адресом из регистра R с его последующим инкрементированием. При расширении инструкций, равном 1, осуществляется запись в регистр B содержимого ячейки ОЗУ с адресом из регистра R с его последующим инкрементированием: **0:A[15:0]<=RAM(R); 1:B[15:0]<=RAM(R); X<=X+1**
20	ARamB	Запись в младшие разряды регистра A содержимого ячейки ОЗУ с адресом из регистра B: **A[15:0]<=RAM(X); X<=B[15:0]**
21	BRamA	Запись в регистр B содержимого ячейки ОЗУ с адресом из младших разрядов регистра A: **B[15:0]<=RAM(X); X<=A[15:0]**
22	REG	Обмен информации между регистрами: При расширении инструкций, равном 0, запись: **A[15:0]<=B[15:0]**. При расширении инструкций, равном 1, запись: **B[15:0]<=A[15:0]**. При расширении инструкций, равном 2, обмен: **B<=>A[15:0]**. При расширении инструкций, равном 3, запись: **R[15:0]<=A[15:0]**. При расширении инструкций, равном 4, запись: **R[15:0]<=B[15:0]**. При расширении инструкций, равном 3, запись: **P[15:0]<=A[15:0]**. При расширении инструкций, равном 4, запись: **P[15:0]<=B[15:0]**

Таблица 2. Общая система команд переходов процессора «КВАРК»

23	jmp+	**Команда безусловного перехода** Увеличение значения счетчика команд на значение, равное операнду из текущей команды
24	jmpx	**Команда безусловного перехода** Присвоение счетчику команд значения операнда из текущей команды

Таблица 2. *Окончание*

25	jmprg	**Команда косвенного перехода** При расширении инструкций, равном 0, — присвоение счетчику команд значения из младших разрядов регистра A. При расширении инструкций, равном 1, — присвоение счетчику команд значения из регистра B
26	jmpRAMX	**Команда косвенного перехода** Присвоение счетчику команд значения ячейки ОЗУ с адресом из операнда текущей команды
27	CM	**Команда условного перехода** При расширении инструкций, не равном 0, при выполнении условия **if(A[15:0]>B[15:0])** — присвоение счетчику команд значения операнда из текущей команды; если условие **if(A[15:0]>B[15:0])** не выполняется, происходит увеличение значения счетчика команд на величину указателя длины команды. При расширении инструкций, равном 0, при выполнении условия **if(A[15:0]>B[15:0])** — инкрементирование значения счетчика команд; если условие **if(A[15:0]>B[15:0])** не выполняется, происходит увеличение значения счетчика команд на удвоенную величину указателя длины команды
28	CL	**Команда условного перехода** При расширении инструкций, не равном 0, при выполнении условия **if(A[15:0]<B[15:0])** — присвоение счетчику команд значения операнда из текущей команды; если условие **if(A[15:0]<B[15:0])** не выполняется, происходит увеличение значения счетчика команд на величину указателя длины команды. При расширении инструкций, равном 0, при выполнении условия **if(A[15:0]<B[15:0])** — инкрементирование значения счетчика команд; если условие **if(A[15:0]<B[15:0])** не выполняется, происходит увеличение значения счетчика команд на удвоенную величину указателя длины команды
29	CE	**Команда условного перехода** При расширении инструкций, не равном 0, при выполнении условия **if(A[15:0]==B[15:0])** — присвоение счетчику команд значения операнда из текущей команды; если условие **if(A[15:0]==B[15:0])** не выполняется, происходит увеличение значения счетчика команд на величину указателя длины команды. При расширении инструкций, равном 0, при выполнении условия **if(A[15:0]==B[15:0])** — инкрементирование значения счетчика команд; если условие **if(A[15:0]==B[15:0])** не выполняется, происходит увеличение значения счетчика команд на удвоенную величину указателя длины команды
30	CN	**Команда условного перехода** При расширении инструкций, не равном 0, при выполнении условия: **if(A[15:0]!=B[15:0])** — присвоение счетчику команд значения операнда из текущей команды; если условие **if(A[15:0]!=B[15:0])** не выполняется, происходит увеличение значения счетчика команд на величину указателя длины команды. При расширении инструкций, равном 0, при выполнении условия **if(A[15:0]==B[15:0])** — инкрементирование значения счетчика команд; если условие **if(A[15:0]==B[15:0])** не выполняется, происходит увеличение значения счетчика команд на удвоенную величину указателя длины команды
31	AEZ	**Команда условного перехода** При расширении инструкций, не равном 0, при выполнении условия **if(A[15:0]!=0)** — присвоение счетчику команд значения операнда из текущей команды; если условие не выполняется, происходит увеличение значения счетчика команд на величину указателя длины команды. При расширении инструкций, равном 0, при выполнении условия **if(A[15:0]!=0** — инкрементирование значения счетчика команд; если условие не выполняется, происходит увеличение значения счетчика команд на удвоенную величину указателя длины команды

Таблица 3. Общая система команд логических и арифметических операций процессора

32	split	**Замена частей внутри регистра** При расширении инструкций, равном 0: **A[31:16]<=A[15:0]; A[15:0]<=A[31:16]**; При расширении инструкций, равном 1: **B[15:8]<=B[7:0]; B[7:0]<=B[15:8]**
33	rotB	Вращение регистра B, замена младшего бита старшим и т.д. **B[15:0]>>>B[0:15]**
34	and	**Побитная логическая операция «И» между регистром B и младшими разрядами регистра A** При расширении инструкций, равном 0: **A[15:0]<=A[15:0]&B[15:0]**; При расширении инструкций, равном 1: **B[15:0]<=B[15:0]&A[15:0]**
35	or	**Побитная логическая операция «ИЛИ» между регистром B и младшими разрядами регистра A** При расширении инструкций, равном 0: **A[15:0]<=A[15:0] |B[15:0]**; При расширении инструкций, равном 1: **B[15:0]<=B[15:0] |A[15:0]**
36	not	**Побитная логическая операция «НЕ»** При расширении инструкций, равном 0: **A[31:0]<=~A[31:0]**; При расширении инструкций, равном 1: **B[15:0]<=~B[15:0]**; При расширении инструкций, равном 2: **A[15:0]<=~B[15:0]**; При расширении инструкций, равном 3: **B[15:0]<=~A[15:0]**
37	xor	**Побитная логическая операция «исключающее ИЛИ»** При расширении инструкций, равном 0: **A[15:0]<=A[15:0]^B[15:0]**; При расширении инструкций, равном 1: **B[15:0]<=B[15:0]^B[15:0]**
38	add	Операция сложения: **A[31:0]<=A[31:0]+B[15:0]**
39	sub	Операция вычитания: **A[31:0]<=A[31:0]−B[15:0]**
40	mlt	Операция умножения: **A[31:0]<=A[15:0]*B[15:0]**
41	div	Операция деления: **A[15:0]<=A[15:0]/B[15:0]**
42	incx	**Сложение с константой** При старшем бите **Operand[15]==1**: **A[31:0]<=A[31:0]+Operand[14:0];** При старшем бите **Operand[15]==0**: **B[15:0]<=B[15:0]+Operand[14:0]**
43	decx	**Вычитание константы** При старшем бите **Operand[15]==1**: **A[31:0]<=A[31:0]−Operand[14:0];** При старшем бите **Operand[15]==0**: **B[15:0]<=B[15:0]−Operand[14:0]**

программ применяется память с шиной данных равной или большей, чем программное слово процессора, другим важным условием является работоспособность памяти на частоте ядра. Однако целью разработки является оптимизация для определенной элементной базы, поэтому КПП включен в структуру и позволяет подключать до трех 8-битных ПЗУ типа 1645РТ2У. КПП поддерживает несколько режимов работы с памятью.

Первый режим — память программ функционирует в режиме параллельного хранения. Из каждой ИМС одновременно считывается по байту, которые объединяются контроллером памяти программ и образуют командное слово. Режим параллельного хранения за одно обращение к памяти позволяет считать все командное слово, что обеспечивает работу на максимальной скорости для ИМС памяти. Данный режим обеспечивает максимально доступный объем памяти программ.

Второй режим — работа с тремя ИМС памяти в мажоритарном режиме «2 из 3». В каждой ИМС ПЗУ хранятся целиком командное слово в трех соседних ячейках, данные с которых последовательно считываются и проходят через мажоритарный элемент вместе с данными из других ИМС памяти. Мажоритарный режим обеспечивает более

Таблица 4. Общая система команд работы с прерываниями процессора

44	softint	Вызов программного прерывания
45	intcomp	Команда выхода из прерывания
46	intoff	Запрет вызова прерываний
47	inton	Разрешение прерываний
48	ext_rom	Переход на выполнение инструкции из периферийной памяти
49	int_rom	Переход на выполнение инструкции из памяти программ
50	wait	Переход в режим ожидания внешнего события
63	delayA	Вызов программируемого ожидания, равного числу тактов из регистра A, во время ожидания продолжается работа основной последовательности (обработка прерываний доступна). При вызове прерывания при активном ожидании счетчик не останавливается, и при выходе из прерывания, если счетчик успел обнулиться, сразу сработает переход на следующую команду; если счетчик еще не обнулен, то операция ожидания будет продолжена: **delay=CLK*A[31:0] (interrupts available)**

низкую скорость считывания программы по сравнению с режимом параллельного хранения и использует только треть доступной памяти программ, однако повышает надежности системы.

Третий режим КПП — режим одной ИМС ПЗУ. В единственной ИМС памяти хранятся все командное слово в виде 3 байт в соседних ячейках, аналогично с мажоритарным режимом. Данные считываются последовательно и КПП формируют командное слово. Скорость работы в данном режиме равна предыдущему, но не обеспечивается повышение надежности системы. Третий режим является самым экономичным и позволяет сократить число корпусов ИМС ПЗУ.

КПП реализует дополнительный режим, который активируется при программном запросе на переход к выполнению программы из внешней памяти, при этом КПП формирует командные слова из данных, полученных по периферийной шине от контроллера внешней памяти. Дополнительный режим проигрывает всем остальным по скорости, так как для формирование командного слова необходимо выполнение длительной процедуры по считыванию 3 байт из внешней, периферийной памяти. Данный режим позволяет расширить пространство доступной памяти программ до любого необходимого объема. Использование указателя длины команды позволяет оптимизировать использование ПЗУ, так как ядро автоматически будет учитывать смещение адресации команд в памяти.

Результаты испытаний процессора

Ядро процессора отдельно проходило проверку в различных конфигурациях и под различные семейства ПЛИС с тестовой программой, использующей набор инструкций облегченного режима. Усредненные результаты представлены в табл. 5.

Из результатов, приведенных в табл. 5, видно, что если брать за основу ПЛИС 5578ТС034, то можно говорить об оптимальном варианте ядра

Таблица 5. Результаты компилирования ip-ядра процессора СМК

Варианты		ПЛИС			
		FLEX	Cyclone III	Cyclone IV	MAX II
Облегченный	объем (элементов)	1930	860	868	870
	MIPS	14,2	38,2	35	28,6
Оптимальный	объем (элементов)	**2752**	1310	1379	1370
	MIPS	**12**	36,8	35	24,3
Максимальный	объем (элементов)	3440	1760	1731	—
	MIPS	5,4	30	27,3	—

(полный набор команд, но без аппаратного умножения и деления), для ПЛИС 5576ХС6Т лучше использовать уменьшенный набор команд, так как оптимальный вариант занимает больше 90 % объема ИМС.

Описание структуры синтезируемого контроллера

В структуру микроконтроллера кроме непосредственно ядра и сильно сопряженных с ним блоков входят разработанные IP-модули различных систем, в основном это модули подключаемый по шине периферии. Структурная схема СМК представлена на рис. 3.

На структурной схеме СМК, кроме ранее рассмотренных блоков ядра, контроллеров прерываний и памяти программ присутствуют периферийные блоки, отвечающие за связь с внешним окружением: контроллеры периферийных устройств, интерфейсов, вводов/выводов и внешней памяти. Другая группа блоков осуществляет ряд внутренних операций, к таким модулям относятся сторожевой таймер, устройство запуска и блок таймеров-счетчиков. Каждый таймер-счетчик из блока может генерировать сигналы прерывания для ядра процессора, а также выводить сигнал в том числе ШИМ на внешние выводы ПЛИС. Сериализатор ШП является специальным модулем, преобразующим параллельный формат ШП в последовательный, и служит для масштабирования периферийных устройств, подключенный к ядру, путем разворачивания ШП и продолжения ее в соседней ИМС ПЛИС. Такой подход сокращает количество межкристальных связей, а при необходимости повышения надежности связи существует возможность мажоритирования последовательных каналов сериализатора ШП.

В структуре СМК существует две основные шины — это шина периферии (ШП) и шина флагов и прерываний (ШФП). Взаимодействие с периферийными устройствами происходит при операциях чтения и записи по ШП. Каждое устройство ШП имеет уникальный адрес и при обращении к нему на следующем такте выставляет на своем выводе результат запроса, запись в периферийные устройства происходит по адресу записи и сигналу разрешения записи. Все взаимодействия по ШП являются синхронными с сигналом тактирования ядра. Прерывания могут генерировать все внутренние блоки — от контроллеров интерфейсов до сторожевого таймера. Сторожевой таймер является устройством, которое имеет программное управление,

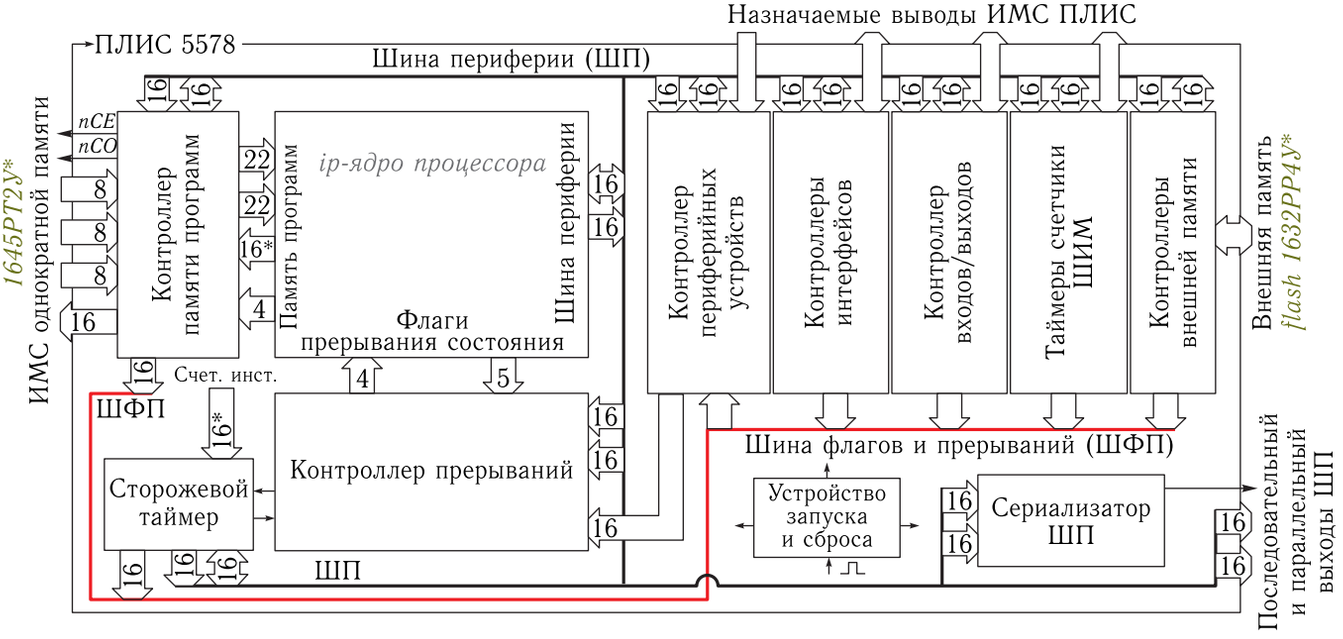

Рис. 3. Структура синтезируемого микроконтроллера

и его необходимо предварительно активировать и настроить из программы, так как он является периферийным устройством со своим уникальным адресом. Основным настраиваемым параметром для сторожевого таймера является максимальное время отсутствия импульсов от счетчика инструкций ядра процессора. Сторожевой таймер позволяет при обращении к нему считать информацию о ранее детектированных нештатных ситуациях и значение счетчика команд, при котором произошло последнее зависание программы.

Взаимодействие с другими модулями, подключенными к ШП, построено аналогичным образом. Модули необходимо инициализировать, отправив им конфигурационные пакеты, содержащие в том числе команду активации. Некоторые модули занимают несколько адресов в области адресов ШП, что связано с тем, что они работают в разных

Таблица 6. Исходный набор периферии СМК на ядре «КВАРК»

Модуль	Кол-во	Характеристики	Примечание
Таймер-счетчик	2	16-разрядный счетчик с 10-разрядным предделителем, возможностью установки триггера и формирования однофазного ШИМ	Может являться источником прерываний
ШИМ-контроллер	4	Специализированный ШИМ-контроллер для управления преобразователями различной топологии. Контроллер имеет возможность формирования парафазного ШИМ-сигнала, выбора разрядности от 6 до 16 бит и возможность установки длительности мертвого времени	Максимальная частота 625 КГц
Контроллер ввода/вывода	1	32 ввода и 32 выхода. Вводы могут являться источником прерываний или событий	Все возможности вводов/выводов ПЛИС
Контроллер внешней памяти	1	Предназначен для работы по интерфейсу SPI с микросхемами flash-памяти (1636РР4У). Есть возможность работать на частоте, равной половине тактового сигнала ПЛИС. Поддерживает до трех параллельно подключенных ИМС в режиме «2 из 3»	Может быть масштабирован для увеличения объема подключенной памяти
UART	4	Поддерживает работу в полнодуплексном режиме. Скорость от 9600 до 921 600. Возможность включения проверки на четность и нечетность. Приемник и передатчик независимы	Может являться источником прерываний
I2C	2	Поддерживает работу с частотой 100 кГц или 400 кГц. Функционирует в режиме мастера. Возможность создания непрерывной сессии с устройством	Может являться источником прерываний
SPI	2	Скорость соединения задается из программы. Максимальная частота равна половине частоты тактирования модуля	Может являться источником прерываний
IWCC	2	Интерфейс с кодированием по стандарту IEEE 802.3. Разработан для высоконадежного межкристального и межблочного соединения на расстояниях до 30 м. Может использоваться для соединения с КПА по витой паре с возможностью гальванической трансформаторной развязки. Доступные скорости 1–10 Мбит/с. Пакеты по 16 бит. Аппаратная реализация CRC8. Длина сессии не ограничена.	Может являться источником прерываний при отправке или приеме
SWI	1	Интерфейс для внутриплатных соединений ИМС ПЛИС по одному проводу на скорости до 1,8 Мбайт/с. Самотактирующийся, пакеты по 16 бит	Может являться источником прерываний

режимах и нуждаются в нескольких конфигурационных пакетах, так как содержат много настраиваемых параметров. Все интерфейсные модули имеют несколько адресов.

При формировании контроллера была разработана развитая структура периферийных устройств, избыточная для целевой ИМС ПЛИС. Данное решение позволяет, отключая не используемые в текущем проекте модули, получать оптимальную структуру, которая способна выполнять необходимые функции и конфигурироваться в объеме доступных логических вентилей ПЛИС. В качестве первоначального шаблона разработан СМК с набором периферийных устройств, представленным в табл. 6, для него разработана таблица адресов, что позволяет каждый раз при изменении состава модулей СМК не заниматься распределением адресного пространства ШП.

После оптимизации для ИМС 5578ТС034 была выработана следующая конфигурация СМК:

1) ip-ядро СМК с «оптимальным» набором команд (61 команда с расширениями);

2) контроллер памяти программ, поддерживающий подключение трех ИМС ПЗУ в мажоритарном режиме и возможность исполнения инструкций из внешней flash памяти;

3) SPI контроллер внешней flash-памяти с мажоритарным входом;

4) ШИМ-контроллер;

5) периферийный таймер-счетчик (есть еще встроенный в ядро);

6) UART — 2 комплекта;

7) SPI — 1 комплект;

8) I2C — 1 комплект;

9) IWCC — 1 комплект (для связи с КПА);

10) выход ШП в параллельном и последовательном виде (через сериализатор);

11) один сторожевой таймер;

12) контроллер входов/выходов — 32 входа/16 выходов;

13) устройство запуска;

14) контроллер прерываний с памятью запросов.

Результаты компиляции СМК в указанной конфигурации представлены в табл. 7. В качестве аналога 5578ТС034 выбрана ПЛИС EPF10K100EBC356-3. Для сравнения были скомпилированы проекты под ИМС: EPF10K100EBC356-1; EP4CE40F29C6; EP3C16F484C6. Для всех вариантов было проведено моделирование с учетом задержек распространения сигналов.

Таблица 7. Результаты компилирования СМК

Параметр	ПЛИС			
	EPF10K100 EBC356-3	EPF10K100 EBC356-1	EP3C16F 256C6	EP4CE40 F29C6
Объем	4944 (99 %)	4944 (99 %)	3175 (6 %)	3200 (3 %)
Производительность	100 %	172 %	540 %	525 %
Память	67 %	67 %	6 %	3 %

Из полученных результатов видно, что для 5578ТС034 данный вариант конфигурации занимает почти весь объем логических элементов, однако остается область неиспользованной памяти. Ее не удается синтезировать целым сектором с единым адресом для 16-битного слова, однако можно создать дополнительный буфер под нужды ядра или периферии. В данной комплектации ОЗУ равно 2048 × 16 бит. Для самой медленной ИМС семейства FLEX максимальная частота, полученная во временном анализаторе, составляет 25 МГц, что значительно уступает другим семействам. Однако предполагаемая память программ по ТО имеет время установления истинных данных до 100 нс, что делает именно ее узким местом системы в плане скорости. Кроме того, архитектура ЦП дает дополнительное время на прохождение сигнала, что позволило пройти моделирование на удвоенной частоте для EPF10K100EBC356-3.

Важной особенностью конфигурации СМК является полностью синхронный дизайн, независимый от аппаратных особенностей ПЛИС, это обеспечивает возможность переноса проекта между любыми семействами ИМС. Важны только доступный объем ОЗУ и количество логических элементов.

На базе данной конфигурации СМК под ПЛИС 5578ТС034 разработан вариант типовой системы управления, структурная схема которой представлена на рис. 4. Устройства со звездочкой на рис. 4 необязательны для работы СМК и приведены как пример использования возможностей СМК.

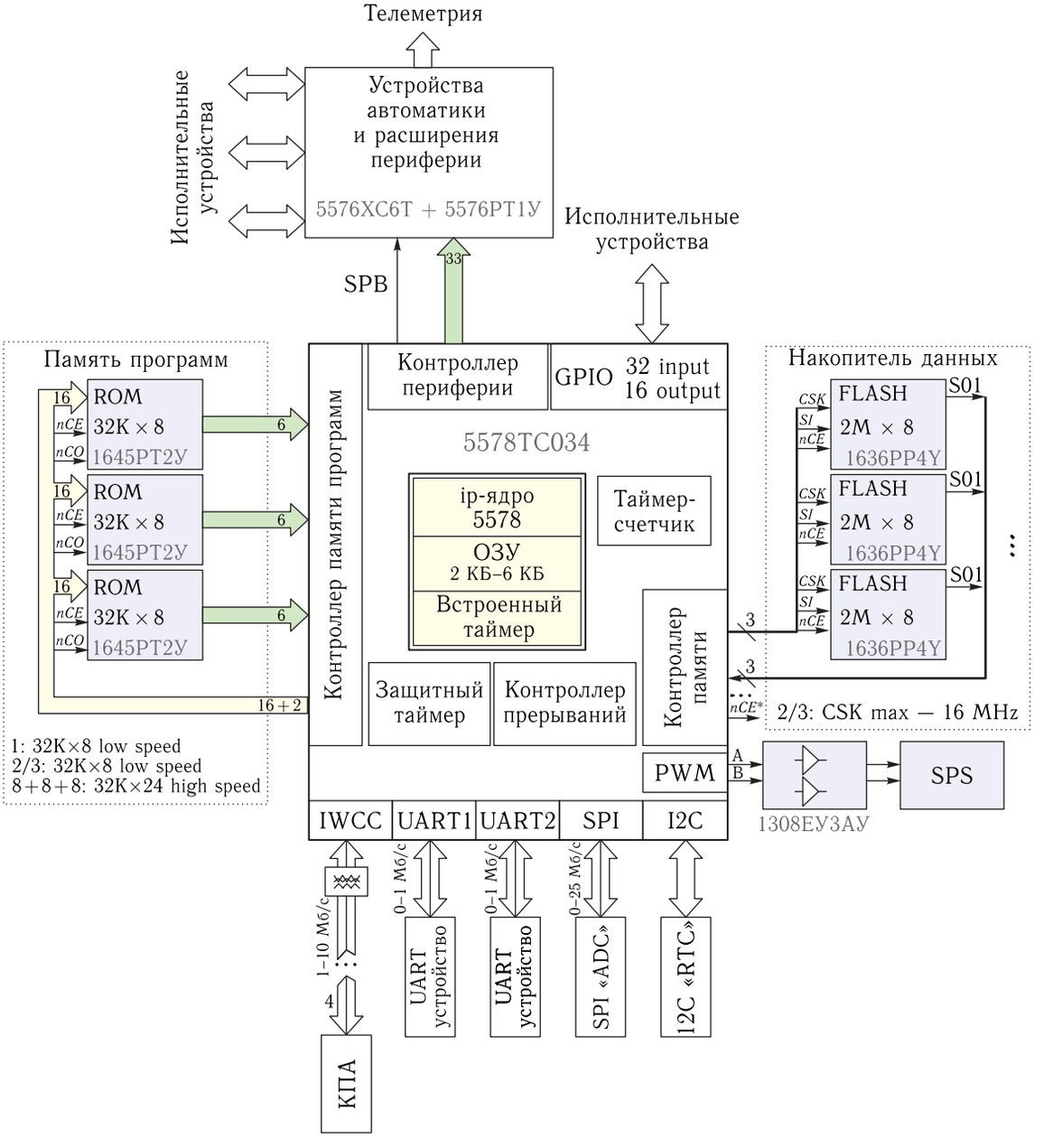

Рис. 4. Структура СУ синтезируемого микроконтроллера

Процесс отладки структуры для 5578ТС034 осуществляется по JTAG и проводится до однократного программирования ИМС. Для отладки программы процессора можно использовать любую flash память с параллельным интерфейсом.

Внешние выводы ИМС жестко не привязаны и могут назначаться после трассировки печатной платы, однако существуют системные выводы, которые переназначить невозможно, при трассировке печатной платы именно им стоит уделить особое внимание.

Разработанная система управления имеет следующие характеристики:

- 32 Кбайт однократной памяти программ, работающей в мажоритарном режиме;
- 2 Мбайт внешнего накопителя на базе flash-памяти, работающей в мажоритарном режиме;

- 4 Кбайт ОЗУ на базе внутренней памяти ИМС ПЛИС5578ТС034;
- работа на частоте до 25 МГц; 16-, 32-битная архитектура ядра;
- возможность масштабирования посредством подключения дополнительных модулей по ШП;
- наличие стандартных и специальных интерфейсов;
- наличие специальных блоков.

Выводы

Показана возможность создания полноценного микроконтроллера на базе отечественных ПЛИС типа 5578ТС034 и более емких ИМС. Разработаны IP-модули периферийных устройств и некоторых интерфейсов. Предложен вариант создания системы управления с использованием разработанного контроллера, практическое применение разработанной СУ. В дальнейших разработках планируется увеличение функциональных возможностей СМК посредством оптимизации IP-модулей и добавления новых.

Примечание

Результаты данной работы были применены в разработке «Система генерации лазерного излучения диодная» [4] совместно с РКК «Энергия».

Список литературы

1. *Никитин А. А.* Реализация радиационно-стойкого кодирования в рамках межкристальной связи систем, состоящих из нескольких программируемых логических интегральных схем // Космическая техника и технологии, 2018, № 4 (23). С. 100–110.
2. *Чекмарев С. А.*, *Вергазов М. Ю.*, *Лукин В. А. и др.* Моделирование бортового компьютера на базе открытых IP-блоков для малых и сверхмалых космических аппаратов // Вестник Сибирского государственного аэрокосмического университета имени академика М. Ю. Решетнева. Красноярск, 2011, вып. 2 (35). С. 141–146.
3. АО «Воронежский завод полупроводниковых приборов — Сборка». Каталог изделий 2020 г. http://www.vzpp-s.ru/production/catalog.pdf (Дата обращения 11.02.2021).
4. АКПС.441372.120ПЗ. «Система генерации лазерного излучения диодная». Пояснительная записка. Самара: [б.и.], 2020. 132 с.