ISSN 0868–5886 НАУЧНОЕ ПРИБОРОСТРОЕНИЕ, 2022, том 32, № 4, с. 88–106

РАЗРАБОТКА ПРИБОРОВ И СИСТЕМ

УДК 629.78

© К. И. Сухачёв, К. Е. Воронов, А. С. Дорофеев, Д. А. Шестаков, А. А. Артюшин, 2022

РАЗРАБОТКА ВЫСОКОПРОИЗВОДИТЕЛЬНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ НА БАЗЕ IP-ЯДРА ДЛЯ КОСМИЧЕСКОЙ НАУЧНОЙ АППАРАТУРЫ

В статье представлен результат разработки и реализации универсального синтезируемого процессорного ядра на интегральных микросхемах ПЛИС отечественного и импортного производства. Показана возможность создания высокопроизводительной вычислительной системы на базе ПЛИС. Приведено описание структуры системы, основного процессорного ядра и сопряженных с ним модулей. Приведена система команд процессора. Представлен процесс разработки программы для синтезируемого контроллера и вариант реализации системы управления на базе разработанного контроллера.

Кл. сл.: ПЛИС, IP-ядро процессора, микроконтроллеры, бортовые системы

ВВЕДЕНИЕ

При разработке научной аппаратуры (НА) для космических исследований приоритетным является использование высоконадежной элементной базы. Часто дополнительным условием является применение отечественной элементной базы. При наличии импортных аналогов, более выгодным и удобным является проведение макетирования и отладки электронных систем именно на них с дальнейшим переводом разработанных систем на электрорадиоизделия (ЭРИ) отечественного производства. В связи с чем возникает необходимость разработки универсального решения, которое можно было бы применять для систем управления, обработки и сбора информации вне зависимости от применяемой элементной базы [1]. Использование стандартных микроконтроллеров (МК) для данных целей затруднительно, т.к. перенос проекта с одного типа МК на другой часто вызывает необходимость существенной доработки программы. А многие стойкие образцы МК снабжены однократной памятью, что существенно усложняет процесс отладки программы. К тому же использование максимально мощного, стойкого к внешним воздействиям МК с развитой периферией часто бывает не выгодно экономически. Для решения данных проблем известны несколько конструкций синтезированных микропроцессоров, описанных в научных статьях [1–8].

В данной статье приводится пример реализации мощной вычислительной системы на базе интегральных микросхем (ИМС) вида ПЛИС — программируемых логических интегральных схем различных производителей. Предложенное решение является универсальным и подходит для реализации на любых микросхемах FPGA (Field-Programmable Gate Array, программируемая пользователем вентильная матрица, ППВМ) как отечественного, так и импортного производства.

ОПИСАНИЕ ГЛАВНОГО ПРОЦЕССОРНОГО ЯДРА

Вычислительная система построена на базе разработанного процессорного IP-ядра, названного "NMR", являющегося процессором с гарвардской архитектурой, и содержит отдельные шину команд, шину данных и несколько шин периферии. Разрядность адреса памяти программ составляет 32 бита, разрядность шины инструкций 16 бит, однако сами инструкции могут иметь различную длину: 16 бит или 32 бита. Разделение некоторых команд на две части предусмотрено архитектурой ядра и не отражается на программе, т.к. система управления процессора по типу команды автоматически определяет смещение адреса. Структура командного слова представлена на рис. 1.

Система команд и архитектура "NMR" разработана специально для работы с внешней памятью программ, организованной в слова по 16 бит, т.к. данная организация памяти является распространенной как в импортной, так и отечественной номенклатуре ЭРИ, в том числе существуют высоконадежные однократно программируемые ПЗУ, например РР45РТ3У. Данная память имеет повышенную стойкость к внешним факторам и организацию 128 К × 16 [9]. Структура IP-ядра "NMR" представлена на рис. 2.

Предполагается, что память программ будет работать на пониженной относительно ядра процессора частоте. Поэтому в организации системы

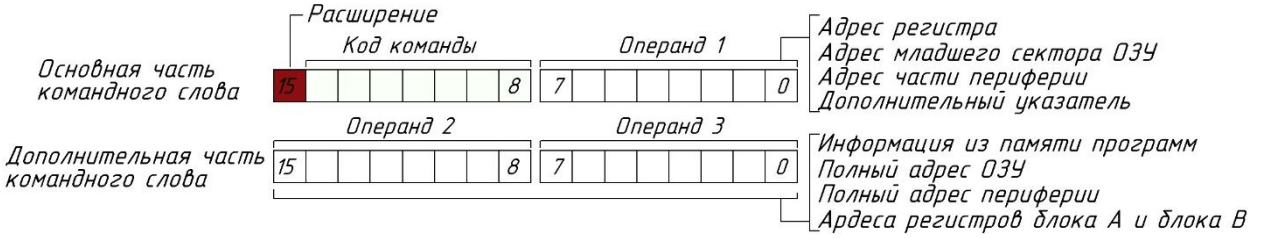

Рис. 1. Структура командного слова

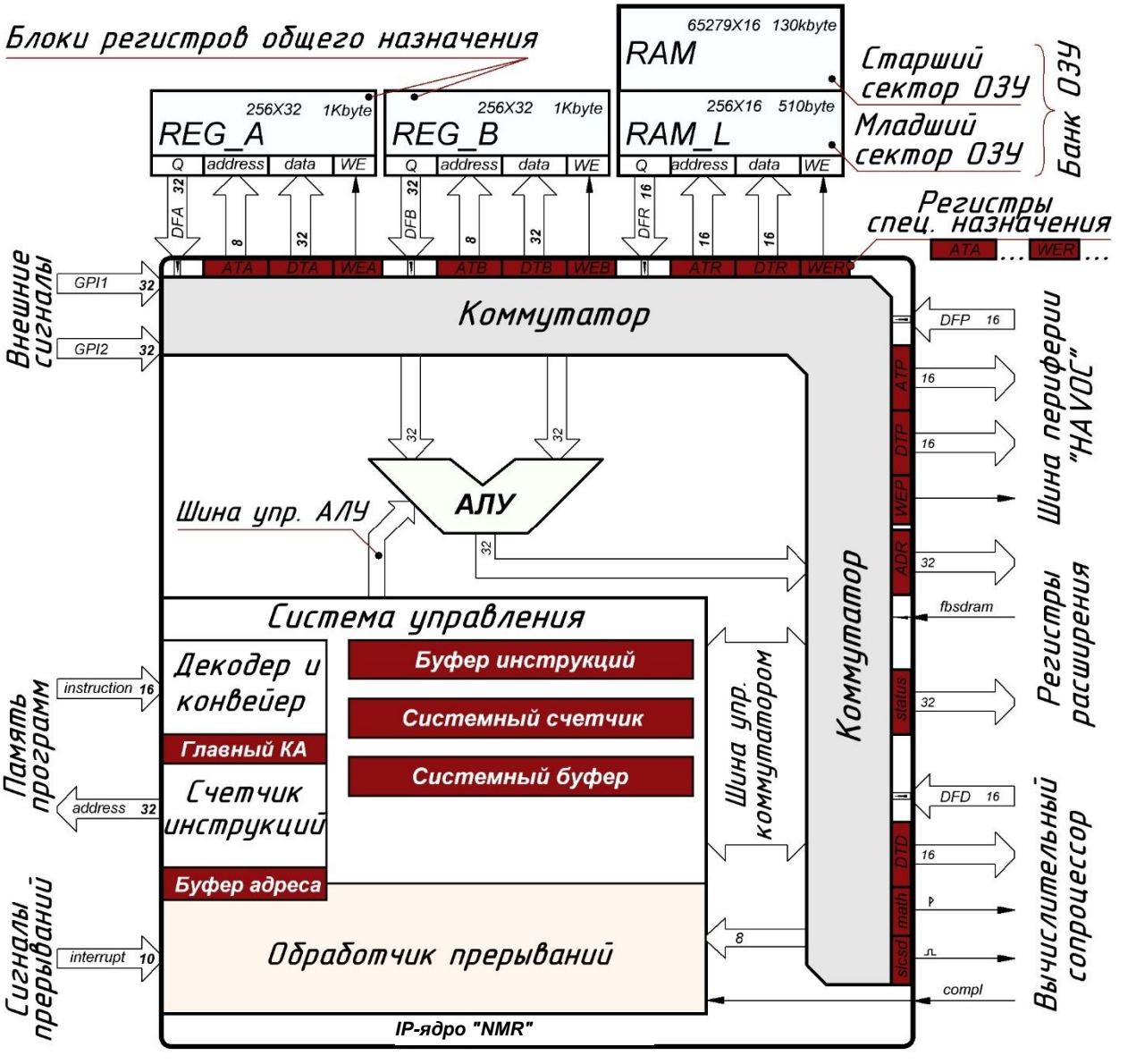

Рис. 2. Структура IP-ядра процессора "NMR"

команд процессора и его внутренней архитектуры реализован подход, позволяющий начать выполнение команды еще до ее полного считывания. Инструкции выполняются на конвейере параллельно

со считыванием частей команд из памяти, что позволяет повысить быстродействие процессора, т.к. память перестает быть узким местом и длительность процессорного такта в среднем соответствует длительности считывания 32-битного командного слова из памяти программ.

В архитектуру "NMR" входит массив регистров общего назначения (РОН), разделенный на два независимых блока REG_A и REG_B. Каждый блок содержит 256 32-битовых регистров. С регистровой памятью возможны все доступные операции, кроме специальных инструкций из системы команд процессора. Система команд "NMR" приведена в Приложении (табл. П1–П5).

Блок работы с оперативной памятью состоит из нескольких частей. Первая часть — для работы с быстрым внутренним банком ОЗУ, который имеет строение, схожее с регистрами общего назначения, и использует внутреннюю ОЗУ ПЛИС, которая имеет разрядность шин данных и адреса по 16 бит, что позволяет адресовать до 131 Кбайт памяти. Первые 256 ячеек ОЗУ (младший сектор) позволяют выполнять большее количество доступных операций, чем с остальным объемом ОЗУ, что позволяет снизить нагрузку на блоки РОН. В "NMR" предусмотрена возможность подключения внешнего ОЗУ через IP-контроллер, для чего предназначены регистры расширения и входные линии внешних сигналов. В зависимости от типа внешнего ОЗУ (SRAM, SDRAM, DDR) IP-контроллер может отличаться.

В системе команд процессора предусмотрены инструкции для работы с динамической памятью, такие как пакетная запись и чтение блоками объемом до 131 Кбайт, остальные функции по взаимодействию непосредственно с ИМС внешнего ОЗУ выполняет IP-контроллер внешнего ОЗУ.

Процессор содержит коммутатор, внешние порты которого буферизированы регистрами специального назначения (РСН). РСН необходимы для работы с блоками регистровой памяти, банком ОЗУ, а также с внешними шинами процессора. В системе команд (Приложение, табл. П1) содержатся инструкции прямого взаимодействия с РСН, что позволяет осуществлять гибкое управление (различные варианты косвенной адресации) и ускорять операции чтения / записи, копирования секторов, потоковое чтение периферии регистровой памяти и ОЗУ. При использовании стандартных операций, связанных с перемещением данных (Приложение, табл. П2), система управления процессором автоматически выполняет все операции с системными регистрами: адресацию, тактирование блоков, выставление и чтение данных с банков памяти. Все периферийные модули подключаются через универсальную адресуемую шину "HAVOC" (Hexadecimal Addressable Vast Outer Connection). Для чтения данных из шины "HAVOC" используется внешний модуль мультиплексора шины, управляемый адресным пространством данной шины. Ядро "NMR" универсально и было испытано на разных ИМС ПЛИС, полученные характеристики ядра сведены в таблицу и представлены в Приложении (табл. П6).

Система команд "NMR" поддерживает условные и безусловные команды переходов, список которых представлен в Приложении (табл. П3).

В систему команд включены команды ***mark***, позволяющие делать аппаратные отметки по ходу исполнения программы, для данной операции доступны как блоки РОН, так и младший сектор ОЗУ. Условные переходы при выполнении условия позволяют увеличить значения счетчика инструкции от 1 до 255. В пропущенной области может находиться ветвь программы, обрабатываемая при невыполнении условия, либо переход на другой сектор памяти, если данной ветви 255 ячеек памяти недостаточно.

Логические и арифметические операции выполняются на 32-битном АЛУ. Доступные на АЛУ операции представлены в Приложении, табл. П4. Ядро "NMR" разработано как универсальное решение для любых ИМС FPGA, поэтому из команд исключены операции аппаратного деления и умножения, т.к. не во всех ИМС есть соответствующие аппаратные блоки. Для выполнения более сложных математических, вычислительных и специальных операций предусмотрена возможность подключения и взаимодействия с вычислительным сопроцессором, который, если ресурсы ИМС это позволяют, может быть размещен как на той же FPGA, так и в отдельной ИМС [10].

Для взаимодействия с сопроцессором существует выделенная скоростная шина, которая позволяет блоками до 256 байт осуществлять пересылку информации между банком ОЗУ "NMR" и ОЗУ сопроцессора. Сопроцессор имеет выделенную линию прерывания. За работу со специальными средствами процессора отвечает последний блок команд (см. Приложение, табл. П5).

Для удобства работы с системой команд в настоящий момент ведется разработка компилятора, сейчас для отладки используется транслятор команд с функцией определения величины смещения при условном переходе, автоматического назначения РОН и навигации по памяти программ. Пример кода и этапы его преобразования представлены в табл.

Программа состоит из объявления параметров памяти, обнуления необходимых РОН, инициализации блоков периферии, в том числе контроллера прерывания. Далее запускается бесконечный цикл с периодической отправкой на внешний порт инкрементируемой переменной.

Табл. Этапы преобразования программы

Исходная программа	Команды процессора	Данные в MIF-файле
#memory size: 8192 #first address: 1024 #interrupt file: C:\NMR\inter1.txt *// zeroing the register, optional:* A0<=16'd0 A0> A0<=16'd0 *// zeroing and writing regB0:* B0<=16'd0 B0> B0<=16'd100 *// zeroing and writing regA1:* A1<=16'd0 A1> A1<=16'd200 *// peripheral module configuration:* PRH41<=16'd25 PRH42<=16'd25 PRH40<=A0 *// activation of vector "1" interrupt:* PRH1<=16'd1 *// naming, optional:* name A0 as counter; name B0 as limit; let PRH40 as SINT; let PRH65535 as outport1; *// program:* pointA: while (counter<limit){ outport1=counter; counter++; } A0<=16'd0 goto pointA; *// sectop program at 8000:* address>>8000 delayA1 SINT=counter; intoff	A0<=16'd0 A0> A0<=16'd0 B0<=16'd0 B0> B0<=16'd100 A1<=16'd0 A1> A1<=16'd200 PRH41<=16'd25 PRH42<=16'd25 PRH40<=A0 PRH1<=16'd1 marck RAM1 marck RAM2 if(A0<B0)up4 PRH65535<=A0 A0++ jamp RAM2 A0<=16'd0 jamp RAM1 address>>8000 delayA1 PRH40<=A0 intoff	0 : 1024 1 : 8000 2 : 8100 3 : 1024 … 1022 : 1024 1023 : 7424 1024 : 0 1025 : 7936 1026 : 7424 1027 : 0 1028 : 7680 1029 : 0 1030 : 8192 1031 : 7680 1032 : 100 1033 : 7425 1034 : 0 1035 : 7937 1036 : 7425 1037 : 200 1038 : 7209 1039 : 25 1040 : 7210 1041 : 25 1042 : 4352 1043 : 40 1044 : 7169 1045 : 1 1046 : 10753 1047 : 10754 1048 : 12032 1049 : 4 1050 : 4352 1051 : 65535 1052 : 16384 1053 : 11522 1054 : 7424 1055 : 0 1056 : 11521 1057 : 0 … 8000 : 32001 8001 : 4352 8002 : 40 8003 : 32515

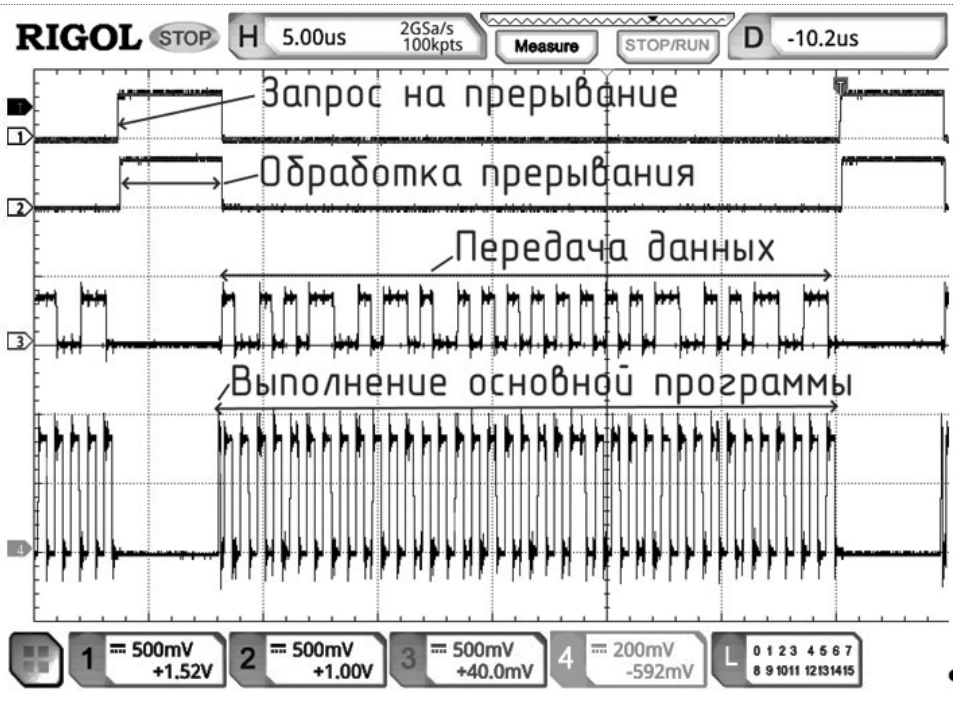

Рис. 3. Осциллограммы исполнения тестовой программы из табл.

Процесс прерывается по сигналу о готовности от модуля передатчика, который захватывает значение переменной и начинает отправку, по завершении которой вызывает прерывание. Программа прерывания описана по адресу начиная с 8000-й ячейки.

На рис. 3 и 4 показаны осциллограммы выполнения программы из табл. на тестовой плате с EP3C25 на частоте ядра и периферии 50 МГц. На рис. 3 представлен полный цикл, в котором выполнение основной программы (инкремент и вывод переменной из РОН в шину периферии (ШП). Сигнал 4 на рис. 3 — младший выводимый разряд переменной *counter*. Проверка условия и условный переход, инкремент одного из РОН, вывод значения в ШП и прыжок на проверку условия (основная программа) занимают примерно 260 нс. На рис. 4 показана реакция на прерывание и разброс времени реакции.

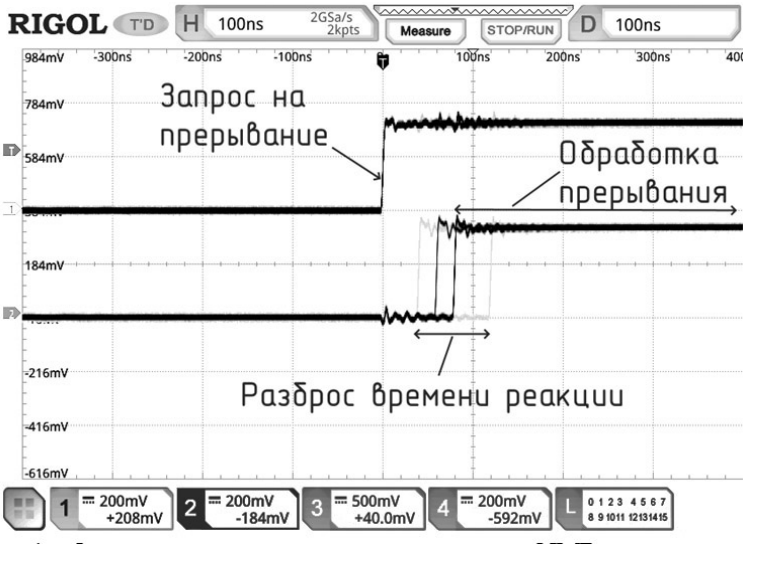

Рис. 4. Отладочные комплекты систем на ядре "NMR" с использованием разных ИМС FPGA

Система управления на базе процессора "NMR"

Для настройки и испытания разработаны две отладочные платы (Приложение, рис. П) на базе ИМС ВЗПП 5578ТС034 и ИМС Intel EP3C25, которая также имеет аналог из каталога ПЛИС ВЗПП 5578ТС094 [9]. Для удобства отладки и испытаний в EP3C25, для хранения небольших программ использовалась ROM-память на базе M9K блоков с загрузкой из конфигурационной flash-памяти EPCS16 в начальный момент. Для исполнения больших программ применялась внешняя ИМС ОЗУ с загрузкой из внешней flash сторонним IP-модулем. Для подачи инструкций ядру внутри 5578ТС034 использовалась плата с EP3C25, подключенная шлейфом — шиной памяти программ.

Результаты, представленные в Приложении, табл. П6, для данных ИМС (рис. 3), изначально получены средствами моделирования в среде QuartusII и подтверждены испытаниями. По временно́му анализу QuartusII, разработанное ядро в семействе FPGA Flex может работать на частотах от 23.5 МГц, однако стабильная работа ядра в 5578ТС034 (аналог) была получена на частоте 20 Мгц, что не является критичным, и, скорее всего, данное явление вызвано особенностями печатной платы и применением шлейфового соединения по шине памяти программ. Ядро в ИМС EP3C25 стабильно работало на частоте до 100 МГц (был применен модуль PLL).

По результатам проделанной работы, проведенным виртуальным и реальным испытаниям, а также с учетом ранее разработанных IP-модулей была предложена структурная схема высокопроизводительной системы обработки и управления на отечественной компонентной базе повышенной стойкости. Применялись и компоненты без повышенной стойкости, например накопитель на flash-памяти, было предусмотрено резервирование. Структурная схема разработанной системы представлена на рис. 5.

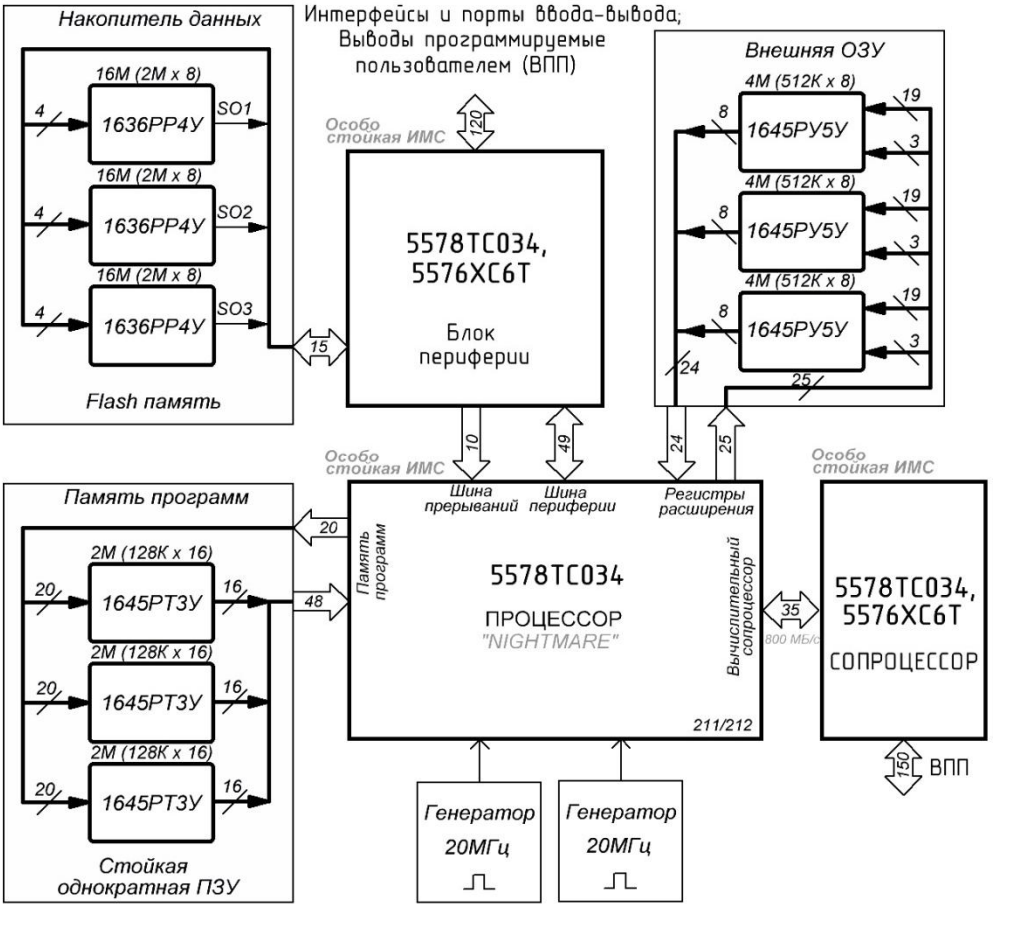

Рис. 5. Структурная схема системы обработки и управления на базе процессорного IP-ядра "NMR"

ВЫВОДЫ

Показана возможность создания производительный вычислительной системы на базе отечественной стойкой элементной базы, в качестве основного компонента выступает ИМС ПЛИС типа 5578ТС034 или более емкие образцы. Предложен вариант создания системы управления и обработки с использованием разработанного процессорного ядра и периферийных модулей. Проведено практическое испытание предложенных решений, результаты которых полностью совпали с результатами моделирования. В дальнейших разработках планируется увеличение функциональных возможностей стандартных МК посредством оптимизации IP-модулей и добавления новых, а также создание программной составляющей среды разработки.

ПРИЛОЖЕНИЕ

Табл. П1. Система команд с регистрами специального назначения (РСН)

Команда	Основ. часть		Доп. часть		Описание команды
	Код	Опр. 1	Опр. 2	Опр. 3	
ATA<=X	1	X	–		Запись числа в системный регистр адреса ATA блока рег. A
ATB<=X	2	X	–		Запись числа в системный регистр адреса ATA блока рег. B
ATR<=X	3	1	X	–	Запись числа в системный регистр адреса ATR ОЗУ
ATP<=X	3	2	X	–	Запись числа в системный регистр адреса ATR ШП "HAVOC"
ATA<=B#	8	#	–	–	Запись в специальные регистры адресов (ATA, ATB, ATR, ATH) содержимого ячеек с адресом [#] из блоков рег. A или рег. B в зависимости от команды
ATB<=A#	9	#	–	–	
ATR<=A#	10	#	–	–	
ATP<=A#	11	#	–	–	
A<=RAM	4	1	–	–	Запись в блок рег. A или рег. B по заранее установленным адресам в регистрах адреса ATA, ATB данных из ОЗУ или ШП "HAVOC" по также заранее установленным адресам в специальных регистрах адреса ATR, ATH соответственно
A<=PRH	4	4	–	–	
B<=RAM	5	1	–	–	
B<=PRH	5	4	–	–	

Табл. П1. *Продолжение*

Команда	Основ. часть		Доп. часть		Описание команды
	Код	Опр. 1	Опр. 2	Опр. 3	
A<=RAM+	4	2	–	–	Запись в блок рег. A или рег. B по заранее установленным адресам в регистрах адреса ATA, ATB данных из ОЗУ или "HAVOC" по также заранее установленным адресам в специальных регистрах адреса ATR, ATH соответственно, значения в которых после записи инкрементируются
A<=PRH+	4	5	–	–	
B<=RAM+	5	2	–		
B<=PRH+	5	5	–	–	
A+<=RAM+	4	3	–	–	Запись в блок рег. A или рег. B по заранее установленным адресам в регистрах адреса ATA, ATB данных из ОЗУ или ШП "HAVOC" по также заранее установленным адресам в специальных регистрах адреса ATR, ATH соответственно, значения всех регистров адреса инкрементируются
A+<=PRH+	4	6	–	–	
B+<=RAM+	5	3	–	–	
B+<=PRH+	5	6	–	–	
RAM<=A	6	1	–	–	Запись в ОЗУ или ШП "HAVOC" по заранее установленным адресам в регистрах ATR и ATH соответственно данных из блоков рег. A или рег. B с также предустановленными адресами в регистрах ATA и ATB
RAM<=B	6	4	–	–	
PRH<=A	7	1	–	–	
PRH<=B	7	4	–	–	
RAM+<=A	6	2	–	–	Запись в ОЗУ или ШП "HAVOC" по заранее установленным адресам в регистрах ATR и ATH соответственно данных из блоков рег. A или рег. B с также предустановленными адресами в регистрах ATA и ATB. Значения ATR или ATH после операции записи инкрементируются
RAM+<=B	6	5	–	–	
PRH+<=A	7	2	–	–	
PRH+<=B	7	5	–	–	

Табл. П1. *Окончание*

Команда	Основ. часть		Доп. часть		Описание команды
	Код	Опр. 1	Опр. 2	Опр. 3	
RAM+<=A+	6	3	–	–	Запись в ОЗУ или ШП "HAVOC" по заранее установленным адресам в регистрах ATR и ATH соответственно данных из блоков рег. A или рег. B с также предустановленными адресами в регистрах ATA и ATB. Значения всех регистров адреса после записи инкрементируются
RAM+<=B+	6	6	–	–	
PRH+<=A+	7	3	–	–	
PRH+<=B+	7	6	–	–	
RAM<=PRH	6	7	–	–	Обмен данными между ОЗУ и ШП "HAVOC" с предустановленными адресами в РСН ATR и ATH с инкрементированием адреса в регистрах ATR и ATH или без в зависимости от команды
PRH<=RAM	7	7	–	–	
RAM+<=PRH	6	8	–	–	
PRH+<=RAM	7	8	–	–	
P=>R(X)	12	–	–	–	Запись из ШП в ОЗУ (код 12) или наоборот (код 13) блока данных объемом X слов, по предустановленным адресам в регистрах ATR, ATH с инкрементом адреса ОЗУ в ATR
R(X)=>P	13	–	–	–	

Табл. П2. Система команд перемещения и записи

Команда	Основ. часть		Доп. часть		Описание команды
	Код	Опр. 1	Опр. 2	Опр. 3	
RAM●<=A#	15	#	●	–	Запись в ОЗУ или ШП "HAVOC" по адресу [●] содержимого ячейки [#] блоков рег. A или рег. B в зависимости от команды
RAM●<=B#	16	#	●	–	
PRH●<=A#	17	#	●	–	
PRH●<=B#	18	#	●	–	

Табл. П2. *Окончание*

Команда	Основ. часть		Доп. часть		Описание команды
	Код	Опр. 1	Опр. 2	Опр. 3	
A#<=RAM●	19	#	●	–	Запись в ячейку [#] блоков рег. A или рег. B из адреса [●] ОЗУ или ШП в зависимости от команды
A#<=PRH●	20	#	●	–	
B#<=RAM●	21	#	●	–	
B#<=PRH●	22	#	●	–	
B#<=A●	23	●	#	–	Запись в ячейку [#] блока рег. B из ячейки [●] блока рег. A
A#<=B●	24	●	#	–	Запись в ячейку [#] блока рег. A из ячейки [●] блока рег. B
RAM#<= X	27	#	X	–	Запись ОЗУ, ШП "HAVOC" или блоки рег. A или рег. B с адресом [#] числа ## из памяти программ. Для ОЗУ и ШП доступны только первые 256 адресов. Для 32-битных регистров запись осуществляется в младшие 16 бит
PRH#<= X	28	#	X	–	
A#<= X	29	#	X	–	
B#<= X	30	#	X	–	
A#><	31	#	–	–	Обмен старших 16 бит на младшие в 32-битной ячейке с адресом [#] блоков рег. A или рег. B
B#><	32	#	–	–	
A#<=A●	33	●	#	–	Перемещение внутри блока одного блока РОН: запись в ячейку [#] ячейки [●]
B#<=B●	34	●	#	–	

Табл. П3. Система команд переходов по пространству адреса программ

Команда	Основ. часть		Доп. часть		Описание команды
	Код	Опр. 1	Опр. 2	Опр. 3	
mark A#	40	#	–	–	Сохранение в ячейке [#] блока регистров A или B текущего номера инструкции (значение счетчика инструкций)
mark B#	41	#	–	–	

Табл. П3. *Окончание*

Команда	Основ. часть		Доп. часть		Описание команды
	Код	Опр. 1	Опр. 2	Опр. 3	
mark RAM#	42	#	–	–	Сохранение в ячейке [#] младшего сектора ОЗУ текущего номера инструкции (значение счетчика инструкций)
jamp A#	43	#	–	–	Безусловный переход по адресу из ячейки [#] блоков регистров A или B, а также из ячейки [#] младшего сектора ОЗУ (присвоение счетчику инструкций значения из ячейки [#])
jamp B#	44	#	–	–	
jamp RAM#	45	#	–	–	
if(A#>B●)up X	46	#	●	X	Условный переход. Если условие выполняется, происходит прыжок на X значений вперед, иначе выполнение программы идет по порядку. Для проверки условия берутся ячейки [#] и [●] из РАЗНЫХ блоков регистров A и B соответственно
if(A#<B●)up X	47	#	●	X	
if(A#=B●)up X	48	#	●	X	
if(A#!=B●)up X	49	#	●	X	

Табл. П4. Логические и арифметические команды

Команда	Основ. часть		Доп. часть		Описание команды
	Код	Опр. 1	Опр. 2	Опр. 3	
A#+X	60	#	X	–	Операция: сложение содержимого ячейки [#] блоков регистров A или B и 16 битного числа X из памяти программ
B#+X	61	#	–	–	
A#-X	62	#	–	–	Операция: вычитание содержимого ячейки [#] блоков регистров A или B и 16 битного числа X из памяти программ
B#-X	63	#	–	–	
A#++	64	#	–	–	Инкремент ячейки [#] блока регистров A
B#++	65	#	–	–	Инкремент ячейки [#] блока регистров B
A#--	66	#	–	–	Декремент ячейки [#] блока регистров A

Табл. П4. *Продолжение*

Команда	Основ. часть		Доп. часть		Описание команды
	Код	Опр. 1	Опр. 2	Опр. 3	
B#--	67	#	–	–	Декремент ячейки [#] блока регистров B
A#<=A●andB◊	68	#	●	◊	Логические операции (and; or; xor) между ячейками РАЗНЫХ блоков регистров: ячейка блока регистров A с адресом [●] и ячейка из блока регистров B с адресом [◊]; результат помещается в ячейку [#] блока регистров A, адрес помещения результата может совпадать с адресом [●] операнда блока регистров A, в таком случае исходное значение будет замещено результатом операции
B#<=A●andB◊	69	#	●	◊	
A#<=A●orB◊	70	#	●	◊	
B#<=A●orB◊	71	#	●	◊	
A#<=A●xorB◊	72	#	●	◊	
B#<=A●xorB◊	73	#	●	◊	
A#<=A●+B◊	74	#	●	◊	Математические операции сложения или вычитания между ячейками [◊], [●] РАЗНЫХ блоков рег. A и рег. B; результат операции помещается в ячейку [#] блоков рег. A и рег. B в зависимости от команды
B#<=A●+B◊	75	#	●	◊	
A#<=A●-B◊	76	#	●	◊	
A#<=B●-A◊	77	#	●	◊	
B#<=A●-B◊	78	#	●	◊	
B#<=B●-A◊	79	#	●	◊	
notA#	80	#	–	–	Логическое побитовое отрицание содержимого ячейки [#] блоков рег. A и рег. B в зависимости от команды
notB#	81	#	–	–	
mask(X)AL#	82	#	X		Побитовая маска младших (код 82) или старших (код 83) 16 бит 32-битной ячейки [#] блока регистров A с числом X
mask(X)AH#	83	#	X	–	
mask(X)BL#	86	#	X	–	Побитовая маска младших (код 86) или старших (код 87) 16 бит 32-битной ячейки [#] блока регистров B с числом X
mask(X)BH#	87	#	X	–	

Табл. П4. *Окончание*

Команда	Основ. часть		Доп. часть		Описание команды
	Код	Опр. 1	Опр. 2	Опр. 3	
rotAL#	84	#	–	–	Операция вращения числа в младшей (код 84) и старшей (код 86) области ячейки [#] блока рег. A или всего 32-битового числа ячейки [#] рег. B (код 88)
rotAH#	85	#	–	–	
rotBF#	88	#	–	–	
A#<< X	89	#	0	X	Сдвиг вправо или влево содержимого ячейки [#] блока регистров A или B на число X
A#>> X	89	#	1	X	
B#<< X	90	#	0	X	
B#>> X	90	#	1	X	
<<A#	91	#	–	–	Сдвиг вправо или влево содержимого ячейки [#] блока регистров A или B на 1 разряд
>>A#	92	#	–	–	

Табл. П5. Система дополнительных команд

Команда	Основ. часть		Доп. часть		Описание команды
	Код	Опр. 1	Опр. 2	Опр. 3	
R#+X =>DTD	93	#	X	–	Отправка или получение в/от сопроцессора пакета данных из/в ОЗУ объемом до X слов, начиная с адреса [#] ОЗУ
R#+X <=DTD	94	#	X	–	
adr<=A#	95	#	–	–	Запись в специальный регистр ADR ячейки [#] блока рег. A
A#<=adr	96	#	–	–	Запись в ячейку [#] блока рег. A содержимого регистра ADR
adr+8'd X	97	X	–	–	Увеличение содержимого специального регистра ADR на X
stat<=A#	##	#	–	–	Запись в специальный регистр Status ячейки [#] блока рег. A

Табл. П5. *Окончание*

Команда	Основ. часть		Доп. часть		Описание команды
	Код	Опр. 1	Опр. 2	Опр. 3	
A#<=stat	##	#	–	–	Запись в ячейку [#] блока рег. A содержимого регистра Status
A#<=GPI1	##	#	–	–	Запись данных со входа GPI1 в ячейку [#] блока рег. A
B#<=GPI2	##	#	–	–	Запись данных со входа GPI1 в ячейку [#] блока рег. A
intoff	##	–	–	–	Команда выхода из прерывания
sleep	##	–	–	–	Остановка процессора, запуск процессора возможен по сигналам прерывания
delayA255	##	#	–	–	Аппаратная задержка на кол-во тактов из ячейки [#] рег. A

Табл. П6. Характеристики ядра "NMR" в разных ИМС ПЛИС

Параметр	ПЛИС					
	5578 TC034	EPF10K100-EBC356-3	EP2C8F-256C6	EP3C16-U484C6	EP3C40-F484C6	EP355F-780C8
Занимаемый объем	4807 (96%)	4807 (96%)	3519 (43%)	3540 (23%)	3535 (9%)	3560 (6%)
Занимаемая память [бит]	32768 (67%)	32768 (67%)	147456 (85%)	81920 (82%)	147456 (13%)	278528 (12%)
Кол-во РОН	512	512	512	512	512	512
Объем ОЗУ	2 КБ	2 КБ	16 КБ	8 КБ	16 КБ	32 КБ
Операций в секунду	6.4 MIPS	7.5 MIPS	25.4 MIPS	47.0 MIPS	45.3 MIPS	37.2 MIPS
Макс. частота	20 МГц	23.5 МГц	79 МГц	149 МГц	141 МГц	116 МГц
Система команд	108 инстр.	108 инстр.	полная	полная	полная	полная
Доп. проц.	внешний	внешний	Akeron	Akeron	Akeron + Sip CPU	Akeron + Sip CPU
Время отклика на прерывание, не более	300 нс	255 нс	76 нс	40 нс	42 нс	51 нс

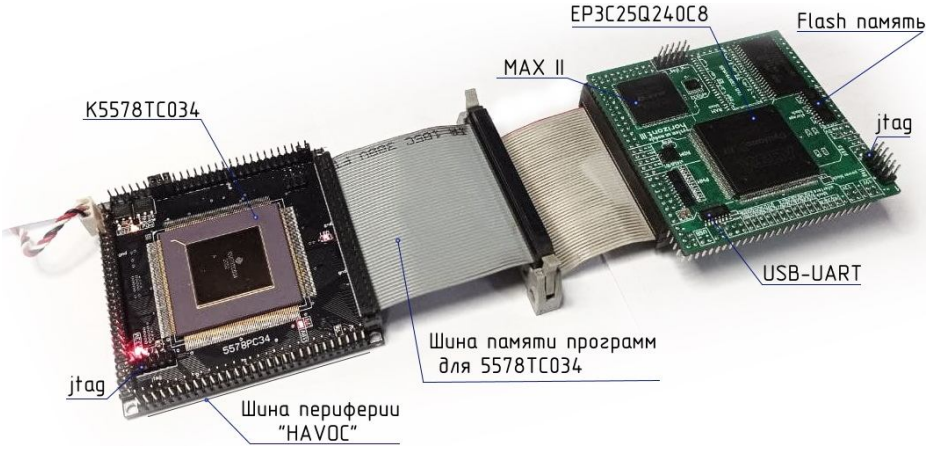

Рис. П. Отладочные комплекты систем на ядре "NMR" с использованием разных ИМС FPGA

СПИСОК ЛИТЕРАТУРЫ

1. *Воронов К.Е., Сухачев К.И., Воробьев Д.С.* Разработка бортового модуля управления на базе вычислительного IP-ядра // Ракетно-космическое приборостроение и информационные системы. 2021. Т. 8, № 1. С. 24–38. DOI: 10.30894/issn2409-0239.2021.8.1.24.38
2. *Haskell R.E., Hanna D.M.* A VHDL--Forth Core for FPGAs // Microprocessors and Microsystems. 2004. Vol. 28, is. 3. P. 115–125.
3. *Ehliar A., Karlstrom P., Liu D.* A high performance microprocessor with DSP extensions optimized for the virtex-4 FPGa // 2008 International Conference on Field Programmable Logic and Applications. IEEE. 2008. P. 599–602.
4. *Tamagnone M., Martina M., Masera G.* An application specific instruction set processor based implementation for signal detection in multiple antenna systems // Microprocessors and Microsystems. 2012. Vol. 36, is. 3. P. 245–256.
5. *Зотов В.* PicoBlaze — семейство восьмиразрядных микропроцессорных ядер, реализуемых на основе ПЛИС фирмы Xilinx // Компоненты и технологии. 2003. № 30. С. 194–198.
6. *Jesman R., Vallina F.M., Saniie J.* MicroBlaze tutorial creating a simple embedded system and adding custom peripherals using Xilinx EDK software tools. Embedded Computing and Signal Processing Laboratory, Illinois Institute of Technology, 2006.
7. *Atehortúa J.C.B.* Desarrollo e implementación del procesador soft-core LatticeMico32 en una FPGA. 2016. URL: https://1library.co/document/ye9524eq-desarrollo-implementacion-procesador-soft-core-latticemico-fpga.html
8. *Pokale M.S.M., Kulkarni M.K., Rode S.V.* NIOS II processor implementation in FPGA: an application of data logging system // Int. J. Sci. Technol. Res. 2012. Vol. 1, is. 11.
9. АО "Воронежский завод полупроводниковых приборов" (ВЗПП-С). Каталог изделий 2020 г. URL: http://www.vzpp-s.ru/production/catalog.pdf (дата обращения: 01.07.2022).
10. *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.

Самарский национальный исследовательский университет им. С.П. Королева, г. Самара

Контакты: *Артюшин Андрей Алексеевич*, artyushin.aa@ssau.ru

Материал поступил в редакцию 01.09.2022

ISSN 0868–5886 NAUCHNOE PRIBOROSTROENIE, 2022, Vol. 32, No. 4, pp. 88–106

DEVELOPMENT OF A HIGH-PERFORMANCE COMPUTING SYSTEM BASED ON AN IP-CORE FOR SPACE SCIENTIFIC EQUIPMENT

K. I. Sukhachev, K. E. Voronov, A. S. Dorofeev, D. A. Shestakov, A. A. Artyushin

Samara National Research University, Samara, Russian Federation

The article presents the result of the development and implementation of a universal synthesized processor core based on integrated FPGA circuits of domestic and foreign production. The possibility of creating a high-performance computing system based on a FPGA is shown. The description of the structure of the system, the main processor core and associated modules, is described. The system of processor commands is presented. The process of developing a program for a synthesized controller and a variant of implementing a control system based on the developed controller are presented.

Keywords: FPGA, IP processor core, microcontrollers, onboard control systems

INTRODUCTION

When developing scientific equipment for space research, the use of highly reliable element base is a priority. Often an additional condition is the use of a domestic element base. In the presence of imported analogues, it is more profitable and convenient to use them for modeling and debugging electronic systems, with the further transfer of the developed systems to domestically produced EEE parts. In this regard, there is a need to develop a universal solution that could be employed in systems of control, processing, and collection of information, regardless of the element base used [1]. The use of standard microcontrollers (MCs) for these purposes is difficult, since the transfer of the project from one type of MC to another often necessitates a significant refinement of the program. Many resistive MC samples are equipped with a write-once memory, which significantly complicates the program's debugging. In addition, the use of the most powerful MC that has developed a periphery and is resistant to external influences is often not economically profitable [1–8].

This article provides an example of the implementation of a powerful computing system based on integrated circuits (ICs) of the programmable logic device (PLD) type from various manufacturers. The proposed solution is universal and suitable for implementation on any FPGA (Field-Programmable Gate Array) microcircuit of both domestic and imported production.

THE MAIN PROCESSOR CORE DESCRIPTION

The computer system is based on the processor IP core, called NMR, which is a Harvard architecture processor and contains a separate command bus, data bus, and several peripheral buses. The bit length of the program memory address is 32 bits, the bit length of the instruction bus is 16 bits, but the instructions themselves can have different lengths: 16 bits or 32 bits. The division of some commands into two parts is provided by the core architecture and does not affect the program since the processor control system automatically detects the address offset by the command type. The structure of the command word is shown in Fig. 1.

Fig. 1. Command word structure

The NMR command system and architecture are specifically designed for working with external program memory organized into words of 16 bits, since this memory organization is common in both imported and domestic EEE parts, including highly reliable once-programmable ROMs, for example RR45RT3U. This memory has increased resistance to external factors and has an organization of 128 K × 16 [9]. The structure of the NMR IP core is shown in Fig. 2.

Fig. 2. NMR processor IP core structure

It is assumed that the program memory will work at a reduced frequency relative to the processor core. Therefore, in the organization of the processor instruction system and its internal architecture, an approach is implemented that allows a command to start even before it is completely read. The instructions are executed on the pipeline in parallel with reading parts of commands from the memory, which increases

the processor speed since the memory ceases to be a bottleneck and the duration of the processor clock on average corresponds to the duration of reading the 32-bit command word from the program memory.

The NMR architecture includes an array of general-purpose registers (GPRs), divided into two independent units REG_A and REG_B. Each unit contains 256 32-bit registers. All available operations are possible with register memory, except for special instructions from the processor instruction set. The NMR command system is given in the Appendix (Tab. П1–П5).

The RAM block consists of several parts. The first part is to work with a fast internal RAM bank, which has a structure similar to general-purpose registers and uses an internal RAM PLD, which has a data bus and an address width of 16 bits, which allows addressing up to 131 KB of memory. The first 256 RAM cells (low sector) allow one to perform more available operations than with the rest of the RAM, which reduces the load on the GPR units. The NMR provides the ability to connect external RAM via an IP controller, for which expansion registers and input lines for external signals are designed. Depending on the type of external RAM (SRAM, SDRAM, DDR), the IP controller may differ.

The processor command system provides instructions for working with dynamic memory, such as packet writing and reading in blocks up to 131 KB, the rest of the functions for interacting directly with the external RAM IM are performed by the external RAM IP controller.

The processor contains a switch, the external ports of which are buffered by special-purpose registers. SPGs are necessary for working with register memory units, RAM bank, as well as external processor buses. The command system (Appendix, Tab. П1) contains instructions for direct interaction with the SPR, which allows flexible control (various options for indirect addressing) and acceleration of read/write operations, sector copying, and streaming reading of the peripheral of the register memory and RAM). When using standard operations related to data movement (Appendix, Tab. П2), the processor control system automatically performs all operations with system registers: addressing, block timing, setting and reading data from memory banks. All peripheral modules are connected via the Hexadecimal Addressable Vast Outer Connection. To read data from the HAVOC bus, an external bus multiplexer module is used, controlled by the address space of this bus. The NMR core is universal and has been tested on different PLD ICs. The obtained core characteristics are tabulated and presented in the Appendix (Tab. П6).

The NMR command system supports conditional and unconditional transition commands, the list of which is given in the Appendix (Tab. П3).

The command system includes ***mark*** commands that allow one to make hardware marks during the execution of the program. Both the GPR units and the lower RAM sector are available for this operation. Conditional transitions, when the condition is met, increase the statement counter values from 1 to 255. The skipped area may contain a program branch that is processed when the condition is not met, or a jump to another memory sector if 255 memory cells for this branch are not enough.

Logical and arithmetic operations are perfomed on the 32-bit ALU. The operations available on ALU are presented in the Appendix, Tab. П4. The NMR core is designed as a universal solution for any FPGA ICs. Therefore, hardware division and multiplication operations are excluded from the commands, since not all ICs have corresponding hardware blocks. To perform more complex mathematical, computational, and special operations, it is possible to connect to and interact with a computational coprocessor, which, if the IC resources allow, can be placed either on the same FPGA or in a separate IC [10].

To interact with the coprocessor, there is a dedicated high-speed bus that allows blocks of up to 256 bytes to transfer information between the NMR RAM bank and the coprocessor RAM. The coprocessor has a dedicated interrupt line. The last block of commands is responsible for working with the special means of the processor (see Appendix, Tab. П5).

A compiler is currently being developed for the convenience of working with the command system. At present, for debugging, a command translator is used. It has the functions of determining the offset value during a conditional transition, automatic assignment of the GPR, and navigation through the program memory. An example of the code and its transformation steps are shown in Tab.

Tab. Program transformation steps

The program consists of declaring memory parameters, zeroing the required GPRs, initializing peripheral units, including the interrupt controller. Then an infinite loop is started with the periodic sending of an incremented variable to an external port.

The process is interrupted by a ready signal from the transmitter module, which captures the value of the variable and starts sending, upon completion of which it causes an interrupt. The interrupt program is described at the address, starting from the 8000 th cell.

Figs. 3 and 4 show the oscillograms of the program's (from Tab.) execution on the test board with EP3C25 at core and peripheral frequencies of 50 MHz.

Fig. 3 shows the complete cycle in which the main program is executed (increment and output of the variable from the GPR to the peripheral bus (PB).

Fig. 3. Oscillograms of the execution of the test program given in the Tab.

Signal 4 in Fig. 3 is the low output bit of the *counter* variable. Condition check and conditional transition, increment of one of the GPRs, output of the value to the PB, and jump for condition check (main program) take approximately 260 ns. Fig. 4 shows the response to interruption and the response time spread.

Fig. 4. Debug kits of systems on the NMR core using various PLD ICs

NMR processor-based control system

For tuning and testing, two debug boards (Appendix, Fig. П) based on IC VZPP 5578TS034 and IC Intel EP3C25 have been developed. The latter has an analogue of 5578TS094 from the catalog VZPP FPGA [9]. For ease of debugging and testing the EP3C25, ROM memory based on M9K blocks was used to store small programs with a boot from the configuration flash memory EPCS16 at the beginning. To execute large programs, an external RAM IC was used with loading from an external flash by a third-party IP module. To give instructions to the core inside the 5578TS034, an EP3C25 board was used, connected by a loop — the program memory bus.

The results presented in the Appendix, Tab. П6 for the IC data (Fig. 3) were initially obtained by modeling tools in the QuartusII environment and confirmed by tests. According to the QuartusII time analysis, the developed core in the FPGA Flex family can operate at frequencies up to 23.5 MHz, but the stable core operation in 5578TS034 (analog) was obtained at a frequency of 20 MHz, which is not critical. and, most likely, this phenomenon is caused by the features of the printed circuit board and the use of a daisy chain via the program memory bus. The core in the IC EP3C25 worked stably at a frequency of up to 100 MHz (the PLL module was used).

Grounded on the results of the work done, virtual and real tests, as well as taking into account the previously developed IP-modules, a structural diagram of a high-performance processing and control system based on a domestic component supply of increased resistance was proposed. Components without increased resistance were used, for example, a storage device on flash memory. Also, redundancy was provided. The structural diagram of the developed system is shown in Fig. 5.

Fig. 5. Block diagram of the processing and control system based on the IP processor core NMR

CONCLUSIONS

The possibility of creating a productive computing system based on the domestic reliable element supply is shown. The main component is an FPGA IC of type 5578TS034 or more capacious samples. A variant of creating a control and processing system using the developed processor core and peripheral modules is proposed. A practical test of the proposed solutions was carried out, and the results completely coincided with the results of the modeling. In further improvements, it is planned to increase the functionality of standard MCs by optimizing IP modules, adding new ones, and creating a software component of the development environment.

APPENDIX

Tab. П1. Command system with special purpose registers (SPRs)

Tab. П2. Move and write command system

Tab. П3. Program address space navigation command system

Tab. П4. Logical and arithmetic commands

Tab. П5. Additional command system

Tab. II6. Characteristics of the NMR core in different PLD ICs

Fig. II. Debug kits of systems on the NMR core using various PLD ICs

REFERENCES

1. Voronov K.E., Sukhachev K.I., Vorobev D.S. [Development of control module based on a computing IP-core]. *Raketno-kosmicheskoe priborostroenie i informatsionnye sistemy* [Rocket-Space Device Engineering and Information Systems], 2021, vol. 8, no. 1, pp. 24–38. DOI: 10.30894/issn2409-0239.2021.8.1.24.38 (In Russ.).
2. Haskell R.E., Hanna D.M. A VHDL--Forth Core for FPGAs. *Microprocessors and Microsystems*, 2004, vol. 28, is. 3, pp. 115–125.
3. Ehliar A., Karlstrom P., Liu D. A high performance microprocessor with DSP extensions optimized for the virtex-4 FPGa. *2008 International Conference on Field Programmable Logic and Applications*, IEEE, 2008, pp. 599–602.
4. Tamagnone M., Martina M., Masera G. An application specific instruction set processor based implementation for signal detection in multiple antenna systems. *Microprocessors and Microsystems,* 2012, vol. 36, is. 3. pp. 245–256.
5. Zotov V. [PicoBlaze is a family of eight-bit microprocessor cores based on Xilinx FPGAs]. *Komponenty i technologii* [Components and Technologies], 2003, no. 30, pp. 194–198.
6. Jesman R., Vallina F.M., Saniie J. *MicroBlaze tutorial creating a simple embedded system and adding custom peripherals using Xilinx EDK software tools*. Embedded Computing and Signal Processing Laboratory, Illinois Institute of Technology, 2006.
7. Atehortúa J.C.B. *Desarrollo e implementación del procesador soft-core LatticeMico32 en una FPGA*. 2016. URL: https://1library.co/document/ye9524eq-desarrollo-implementacion-procesador-soft-core-latticemico-fpga.html
8. Pokale M.S.M., Kulkarni M.K., Rode S.V. NIOS II processor implementation in FPGA: an application of data logging system. *Int. J. Sci. Technol. Res*., 2012, vol. 1, is. 11.
9. *AO "Voronezhskii zavod poluprovodnikovykh priborov" (VZPP-S). Katalog izdelii 2020*. [Product Catalog 2020]. URL: http://www.vzpp-s.ru/production/catalog.pdf (accessed 01.07.2022). (In Russ.).
10. Nikitin A.A. [Implementation of radiation-resistant encoding within the framework of chip-to-chip interconnections in systems consisting of several field-programmable gate arrays]. *Kosmicheskaya tekhnika i tekhnologii* [Space Engineering and Technology], 2018, no. 4 (23), pp. 100–110. (In Russ.).

Contacts: *Artyushin Andrey Alekseevich,*
artyushin.aa@ssau.ru

Article received by the editorial office on 01.09.2022