Утвержден

ЮФКВ.431282.016РЭ-ЛУ

Микросхема интегральная 1879ВМ6Я

Руководство по эксплуатации

ЮФКВ.431282.016РЭ

Инв.№подл.	Подп. и дата	Взам.инв.№	Инв.№дубл.	Подп. и дата	Справ.№
26969-4	23.03.2020	26969-3			

Содержание

Удостоверен ЮФКВ.431282.016РЭ-УД

Список иллюстраций

Список таблиц

1 Введение в архитектуру микросхемы 1879ВМ6Я

Микросхема интегральная 1879ВМ6Я (далее по тексту – микросхема) представляет собой высокопроизводительный процессор цифровой обработки сигналов.

1.1 Основные отличительные особенности процессоров семейства NeuroMatrix®

Микросхема является дальнейшим развитием семейства отечественных процессоров семейства NeuroMatrix® ЗАО НТЦ "Модуль": Л1879ВМ1 (NM6403), 1879ВМ2 (NM6404) и 1879ВМ4Я (NM6405). Данные процессоры относятся к представителям нового класса векторно-конвейерных DSP. Их отличает высокая производительность на задачах обработки больших потоков данных при относительно небольших аппаратных затратах и малом потреблении питания.

Микросхема представляет собой высокопроизводительный матрично-векторный двухядерный микропроцессор, имеющий оригинальную RISC-архитектуру с элементами VLIW (Very Long Instruction Word), SIMD (Single Instruction Multiple Data) и суперскаляра. Каждое из процессорных ядер состоит из управляющего RISC процессора и векторно-матричного сопроцессора. Один сопроцессор занимается обработкой данных с фиксированной точкой, а другой обрабатывает данные в формате с плавающей точкой.

Отличительными особенностями микросхемы являются:

- Аппаратная поддержка матричных и векторных операций, включающая в себя:
  1) Выполнение двухвекторной АЛУ-операции за один процессорный такт;
  2) Умножение предварительно загруженной матрицы данных (весовых коэффициентов) на вектор данных за один процессорный такт. Данная базовая операция позволяет существенно увеличить число операций умножения с накоплением (MAC), приходящихся на долю одной операции ввода/вывода;
  3) Выполнение на проходе функции насыщения над элементами векторов с целью исключения переполнения при арифметических операциях;
  4) Выполнение операции произвольной коммутации отдельных элементов в векторе и даже отдельных разрядов в любом элементе вектора за один процессорный такт.

  Все перечисленные операции выполняются над векторами, представляющими собой 64-разрядные слова, в которых упакованы данные, представленные в дополнительном коде с фиксированной точкой или 32-разрядные данные в формате с плавающей точкой;
- Программная настройка исполнительных узлов для работы с векторами данных в формате с фиксированной точкой, содержащих требуемое количество элементов требуемой разрядности. В общем случае количество элементов в векторе и их разрядность могут принимать любое значение в пределах от 1 до 64. Единственное ограничение заключается в том, что суммарная разрядность всех элементов каждого вектора должна быть равна 64 бит. Данное свойство позволяет даже в пределах одной задачи варьировать соотношением производительность/точность – повышать производительность процессора за счет снижения точности вычислений и, наоборот, повышать точность за счет снижения производительности;
  - Возможность выполнения векторных, матричных операций, а также операций над комплексными числами за один такт при работе с данными в формате с плавающей точкой одинарной точности;
  - Многотактовый характер векторных команд, которые содержат специальное поле, задающее количество повторений их выполнения (от 1 до 32). Такое решение

позволяет аппаратно поддержать короткие циклы и увеличить плотность программного кода;

- Одновременное выполнение до восьми операций ввода/вывода за один процессорный такт;
- Введение в структуру аппаратной вершины системного стека с целью ускорения процесса возврата из подпрограммы (процедуры обработки прерывания);
- Обеспечение модификации содержимого адресного регистра в первом такте при выполнении многотактных команд ввода/вывода, что позволяет ускорить начало выполнения очередной команды, использующей содержимое данного адресного регистра;
- Введение механизма очередей в конвейер процессора с целью достижения максимальной производительности при работе с банками синхронной памяти, имеющими различную глубину конвейера;
- Уменьшение аппаратной сложности процессорного ядра за счет использования единого адресного генератора при выполнении различных команд ввода/вывода;

Перечисленные выше технические решения позволяют эффективно использовать микросхему в качестве высокопроизводительного процессора цифровой обработки сигналов, ориентированного на решение задач обработки больших потоков данных в реальном масштабе времени (цифровая обработка сигналов, обработка изображений, навигация, связь, эмуляция нейронных сетей и т. д.).

1.2 Основные характеристики и назначение микросхемы

Микросхема представляет собой высокопроизводительный векторно-матричный микропроцессор с оригинальной динамической суперскалярной параллельной архитектурой и сверхбольшими словами команд. Он предназначен для использования в качестве основного или дополнительного процессорного узла в вычислительных системах, интенсивно применяющих цифровую обработку сигналов. Его архитектура позволяет эффективно решать широкий круг задач, включая различные векторно-матричные вычислительные операции, вычисление преобразования Фурье, Адамара и прочих, цифровую фильтрацию, цифровую коммутацию. Микросхема может быть использована в качестве базового элемента при построении многопроцессорных параллельных вычислительных систем.

1.2.1 Характеристики микросхемы

В данном разделе приводятся основные характеристики, дающие представление об особенностях архитектуры и производительности микросхемы:

- Формат обрабатываемых данных:
    1) 32-разрядные скалярные данные;
    2) Вектора данных в формате с фиксированной точкой программируемой разрядности от 1 до 64 разрядов, упакованные в 64-разрядные слова;
    3) Вектора 32-разрядных данных в формате с плавающей точкой, упакованные в 64-разрядные слова (одинарная точность);
    4) 64-разрядные данные в формате с плавающей точкой (двойная точность).
- Команды размером 32-разряда и 64-разряда. Каждая команда может задавать несколько различных операций;
- Адресуемое пространство – 4Г 32-разрядных слов (16 Гбайт);

- Аппаратная поддержка операций умножения вектора на матрицу и матрицы на матрицу, как для данных в формате с фиксированной точкой, так и в формате с плавающей точкой;
- Аппаратная поддержка функции насыщения насыщения для данных в формате с фиксированной точкой;
- Единый поток команд для задания векторных и скалярных операций, операций ввода/вывода;
- Многотактовые векторные команды (возможность одновременного выполнения до пяти векторных команд);
- Одновременное выполнение до восьми операций ввода/вывода за один процессорный такт;
- Производительность для данных в формате с фиксированной точкой (количество операций «умножение с накоплением», выполняемых за один такт):
  - 2 MAC для 32-разрядных данных;
  - 4 MAC для 16-разрядных данных;
  - 24 MAC для 8-разрядных данных;
  - 80 MAC для 4-разрядных данных;
  - 224 MAC для 2-разрядных данных;
- Производительность для 32-разрядных данных в формате с плавающей точкой – 32 FLOP (операций с плавающей точкой) за такт;
- Производительность для 64-разрядных данных в формате с плавающей точкой двойной точности – 8 FLOP за такт;
- Наличие двух независимых RISC-процессорных ядер, одно из которых работает с матрично-векторным сопроцессором для обработки данных, представленных в формате с фиксированной точкой и программируемой разрядностью, а другое управляет матрично-векторным сопроцессором арифметики с плавающей точкой;
- Общий объём внутренней статической памяти 16 Мбит;
- Встроенные системные интеграторы, обеспечивающие доступ процессорных ядер к внутренней и внешней памяти;
- Контроллеры прямого доступа к памяти для эффективного обмена периферийных устройств с памятью, а также для аппаратной поддержки пересылок типа память – память;
- Контроллер внутренних и внешних прерываний;
- Параллельный 32-разрядный интерфейс с внешней памятью типа DDR2 400 МГц;
- Четыре высокоскоростных байтовых коммуникационных порта с пропускной способностью не менее 125 Мбайт/с каждый;
- Интерфейс USB (device);
- Интерфейс SPI с четырьмя сигналами выборки кристалла;
- Восемь последовательных портов ввода/вывода общего назначения;
- JTAG интерфейс (IEEE-1149.1);
- Технология изготовления – 65 нм КМОП;
- Тактовая частота – до 500 МГц.

1.2.2 Области применения микросхемы 1879ВМ6Я

Микросхема может применяться в качестве базового элемента для построения как однопроцессорной, так и многопроцессорной вычислительной системы. Также он может быть использован в качестве сопроцессора для векторно-матричных вычислений в комплексных вычислительных системах под управлением ведущего процессора общего назначения или микроконтроллера.

Основными областями применения микросхемы являются:

- Обработка изображений, включая, различные виды фильтрации и MPEG кодирование и декодирование;
- Обработка широкополосных радиолокационных сигналов, в том числе, различные виды цифровой фильтрации, преобразование Фурье, Адамара и прочее;
- Высокопроизводительная коммутация сигналов;
- Навигация.

1.3 Общая структура микросхемы 1879ВМ6Я

Структурная схема микросхемы приведена на Рисунок 1.1 - Структурная схема .

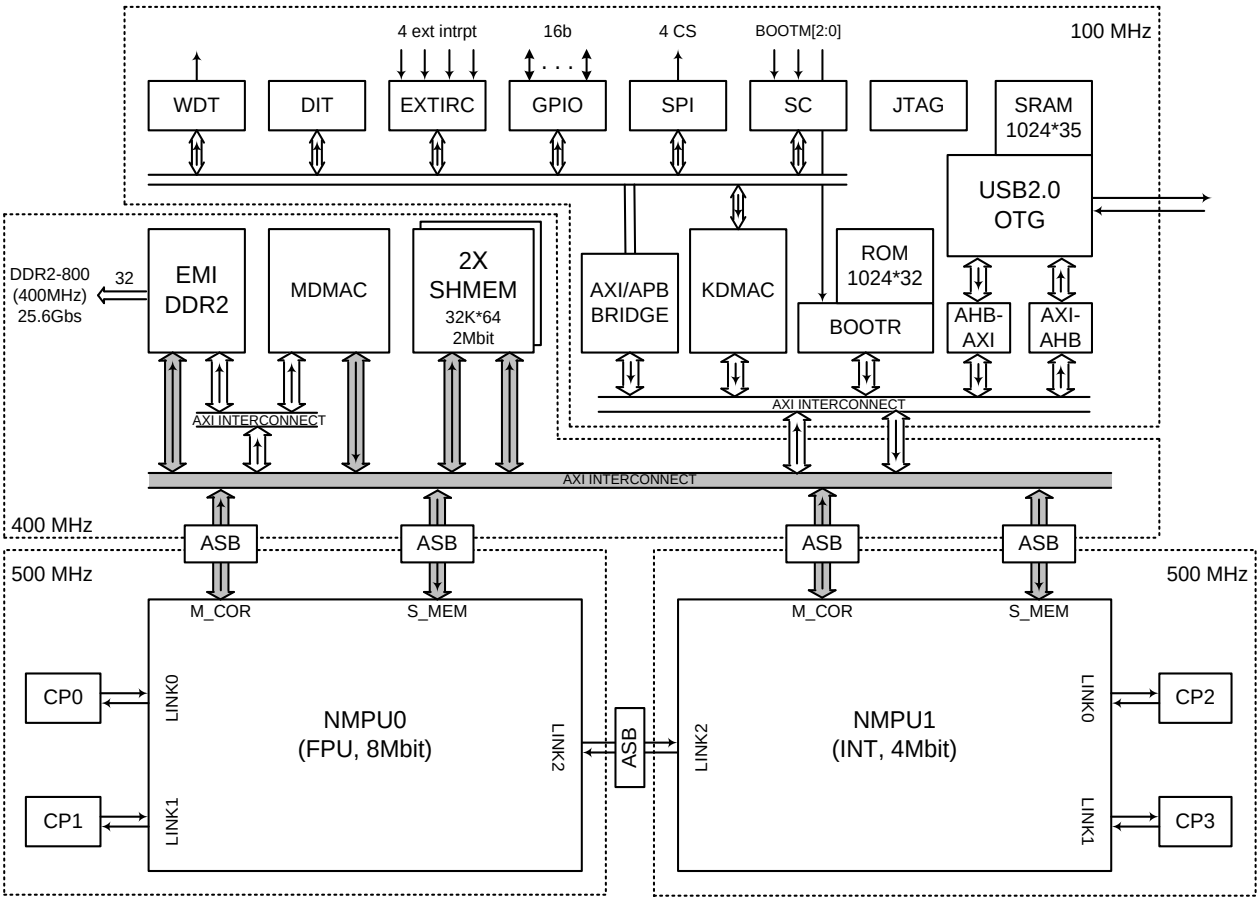


Рисунок 1.1 - Структурная схема микросхемы 1879ВМ6Я

1.3.1 Основные узлы микросхемы

Микросхема состоит из следующих функциональных узлов:

NMPU0 (NeuroMatrix Processor Unit 0) и **NMPU1 (NeuroMatrix Processor Unit 1)** - процессорные системы, каждая из которых состоит из одного и того же RISC-процессорного ядра, но включает в себя разные сопроцессоры: NMPU0 - матрично-векторный сопроцессор арифметики с плавающей точкой, NMPU1 - матрично-векторный сопроцессор для обработки данных, представленных в формате с фиксированной точкой и программируемой разрядностью. Также в состав процессорных систем входит статическая память объёмом 8 Мбит для NMPU0 и 4 Мбит для NMPU1 соответственно, развитая шинная структура с набором коммутаторов, периферийные устройства. Каждая система имеет три 64-разрядных линка (**LINK0 - LINK2**), причём один из линков (LINK2) у них используется для межпроцессорного обмена, а оставшихся два (LINK0 – LINK1) – для обслуживания внешних байтовых коммуникационных портов **CP0 - CP3**. Более подробно процессорные системы NMPU0 и NMPU1 будут описаны в разделе 4.2. Все узлы систем NMPU0 и NMPU1 работают на частоте 500 МГц.

SMEM (Shared Memory Unit) – два банка общей памяти объёмом по 2 Мбит, доступные для каждой процессорной системы: NMPU0 и NMPU1. SMEM работает на частоте 400 МГц.

MDMAC (Memory DMA Controller) – контроллер ПДП, обеспечивающий обмен между внешней и внутренней памятью. Контроллер управляется процессорными системами NMPU0 и NMPU1 и работает на частоте 400 МГц.

EMI (External Memory Interface) – контроллер динамической памяти, реализующий обмен с внешней памятью типа DDR2 SDRAM по 32-разрядной шине данных с частотой 400 МГц, причём за один такт по шине передаются два 32-разрядных данных или одно 64-разрядное данное.

BROM (Boot ROM) – ПЗУ начальной загрузки объёмом 1К*32бит.

BROMC (Boot ROM Controller) – контроллер начальной загрузки BROM в память одной из процессорных систем NMPU0 или NMPU1 в зависимости от режима данной загрузки.

KDMAC (DMA Controller) – контроллер прямого доступа к памяти периферийных устройств. Контроллер управляется процессорными системами NMPU0 и NMPU1.

USB 2.0 DEVICE (Universal Serial Bus 2.0 Device) – контроллер интерфейса последовательной шины USB, соответствующей спецификации 2.0 (Full-speed) и работающий как USB device. Содержит встроенную память 1К*35бит.

WDT (Watch Dog Timer) – блок сторожевого таймера.

DIT (Dual Interval Timers) – блок, содержащий два интервальных таймера.

EXTIRC (External Interrupt Controller) – контроллер внешних прерываний (поддержка до четырех внешних прерываний).

GPIO (General Purpose Input/Output) – блок программируемых вводов/выводов общего назначения. Блок GPIO содержит 16 программируемых вводов/выводов.

SPI (Serial Peripheral Interface) – контроллер синхронного последовательного интерфейса, обеспечивающего обмен с такими интерфейсами как Motorola SPI, National Semiconductor Microwire и Texas Instruments SPI. Поддерживается до четырёх внешних устройств (четыре сигнала выборки кристалла).

SC (System Controller) – системный контроллер, управляющий выбором режима начальной загрузки в зависимости от состояния внешних выводов BOOTM2…BOOTM0.

JTAG (JTAG Interface) – интерфейс с 5-выводным тестовым портом, реализованным согласно стандарту IEEE Std. 1149.1-1990. JTAG интерфейс позволяет тестировать микросхему 1879ВМ6Я как законченное изделие, так и в составе электронной аппаратуры.

Все функциональные узлы микросхемы соединяются шинами трёх типов:

- 64-разрядные шины, позволяющие производить обмен данными на частоте 500 МГц (шины M_COR (MASTER) и S_MEM (SLAVE) процессорных систем NMPU0 и NMPU1);
- 64-разрядные шины, позволяющие производить обмен данными на частоте 400 МГц (затемнённые шины, объединённые шинным коммутатором PROCESSOR INTERCONNECT и работающие согласно стандарту AXI 3.0 фирмы ARM Ltd.);
- 32-разрядные шины, позволяющие производить обмен данными на частоте 100 МГц (не закрашенные шины, объединённые в одну общую периферийную шину - APB(AMBA Peripheral Bus).

Шины первого и второго типа объединяются с помощью специальных асинхронных буферов **ASB (Asynchronous Buffer)**, а второго и третьего – с помощью шинного моста **AXI/APB BRIDGE**. Для работы с контроллером интерфейса последовательной шины USB используется шинный мост **AXI/AHB**.

Здесь и далее шины изображены таким образом, чтобы показать направление как потоков данных, так и адресов. Широкая стрелка отображает направление обмена данными (шину данных), узкая стрелка внутри неё – направление выдачи адреса (шину адреса).

1.3.2 Функциональные выводы

Микросхема имеет 192 функциональных вывода, назначение которых приведено в Таблица 1.1.

Таблица 1.1 - Функциональные выводы СБИС 1879ВМ6Я

Обозначение [1]	Кол-во	Тип[2]	Функциональное назначение
Интерфейс с внешней памятью типа DDR2 (78 выводов)			
DQ0 … DQ31	32	I/O	Шина данных
A0 … A14	15	O	Шина адреса
XCS0, XCS1	2	O	Выборка банков внешней памяти
XWE	1	O	Разрешение записи в память
XRAS	1	O	Строб адреса строки
XCAS	1	O	Строб адреса столбца
DM0 … DM3	4	I/O	Сигналы маскирования данных при записи
DQS0 … DQS3	4	I/O	Сигналы стробов данных
XDQS0…XDQS3	4	I/O	Инверсные сигналы стробов данных
BA0…BA2	3	O	Сигналы выбора банка памяти
ODT0, ODT1	2	O	Сигналы ODT (Memory on-die termination control signal)
CK0, CK1	2	O	Синхросигналы шины
XCK0, XCK1	2	O	Инверсные синхросигналы шины
CKE0, CKE1	2	O	Сигналы разрешения синхросигналов
VREF1 … VREF3	3	AI	Входы напряжения смещения

Продолжение таблицы 1.1

Обозначение [1]	Кол-во	Тип[2]	Функциональное назначение
Коммуникационный порт 0 (13 выводов)			
C0D0 … C0D7	8	I/O	Шина данных
XC0STRB	1	I/O	Строб данных
XC0RDY	1	I/O	Готовность к приёму данных
XC0HOLDI	1	I	Запрос внешнего устройства на передачу данных
XC0HOLDO	1	O	Запрос процессора на передачу данных
C0IS	1	I	Состояние порта после системного сброса
Коммуникационный порт 1 (13 выводов)			
C1D0 … C1D7	8	I/O	Шина данных
XC1STRB	1	I/O	Строб данных
XC1RDY	1	I/O	Готовность к приёму данных
XC1HOLDI	1	I	Запрос внешнего устройства на передачу данных
XC1HOLDO	1	O	Запрос процессора на передачу данных
C1IS	1	I	Состояние порта после системного сброса
Коммуникационный порт 2 (13 выводов)			
C2D0 … C2D7	8	I/O	Шина данных
XC2STRB	1	I/O	Строб данных
XC2RDY	1	I/O	Готовность к приёму данных
XC2HOLDI	1	I	Запрос внешнего устройства на передачу данных
XC2HOLDO	1	O	Запрос процессора на передачу данных
C2IS	1	I	Состояние порта после системного сброса
Коммуникационный порт 3 (13 выводов)			
C3D0 … C3D7	8	I/O	Шина данных
XC3STRB	1	I/O	Строб данных
XC3RDY	1	I/O	Готовность к приёму данных
XC3HOLDI	1	I	Запрос внешнего устройства на передачу данных
XC3HOLDO	1	O	Запрос процессора на передачу данных
C3IS	1	I	Состояние порта после системного сброса
Порты общего назначения (16 выводов)			
GPIO0 … GPIO15	16	I/O	Программируемые порты ввода/вывода
Выводы процессорных систем NMPU0 и NMPU1 (6 выводов)			
U0XNMI	1	I	Вход немаскируемого прерывания NMPU0
U0TIME0	2	I/O	Вход/выход таймера 0 NMPU0
U0TIME1	1	I/O	Вход/выход таймера 1 NMPU0
U1XNMI	1	I	Вход немаскируемого прерывания NMPU1
U1TIME0	1	I/O	Вход/выход таймера 0 NMPU1
U1TIME1	1	I/O	Вход/выход таймера 1 NMPU1
Выводы SPI интерфейса (7 выводов)			
SPICLK	1	O	Синхросигнал SPI интерфейса
SPITXD	1	O	Передаваемые данные
SPIRXD	1	I	Принимаемые данные
SPI_CS0 … SPI_CS3	4	O	Выбор ведомого SPI устройства 0 … 3

Продолжение таблицы 1.1

Обозначение [1]	Кол-во	Тип[2]	Функциональное назначение
Выводы USB2.0 интерфейса full speed device (10 выводов)			
USBCLKSEL	1	I	Выбор источника тактового сигнала
VBUS	1	I/O	Питание 5 В USB
ID	1	I	Идентификатор мини приемника USB
DP	1	I/O	Линия D+ шины USB
DM	1	I/O	Линия D– шины USB
TXRTUNE	1	I/O	Настройка сопротивления передатчика
USB_XI	1	I/O	Вывод подключения внешнего осциллятора 12,0 МГц
USB_XO	1	I/O	Вывод подключения внешнего осциллятора 12,0 МГц или подключения внешнего тактового генератора 12,0 МГц
USBATEST	1	I/O	Тестирование постоянной составляющей тока
DRVVBUS	1	O	Управление внешней схемой генерации питания VBUS (OTG режим)
JTAG интерфейс (5 выводов)			
TDO	1	O(Z)	Выход данных тестового порта JTAG
TDI	1	I	Вход данных тестового порта JTAG
TCK	1	I	Тактовый сигнал тестового порта JTAG
TMS	1	I	Выбор режима тестирования JTAG
XTRST	1	I	Сброс тестового порта JTAG
Общее управление (18 выводов)			
INT0…INT3	4	I	Входы внешних прерываний
BOOTM0…BOOTM2	3	I	Режим начальной загрузки процессора
XRST	1	I	Системный сброс
PCLK_XI	1	I	Опорный тактовый сигнал процессора
PCLK_XO	1	O	Опорный тактовый сигнал процессора
SCLK_XI	1	I	Опорный тактовый сигнал периферии процессора
SCLK_XO	1	O	Опорный тактовый сигнал периферии процессора
PLLBP	1	I	Режим работы в обход встроенных PLL (режим “bypass”)
PLLPD	1	I	Перевод PLL в состояние “power down”
PLLSTNDBY	1	I	Перевод PLL в состояние “stand-by”
IDDQ_EN	1	I	Режим тестирования интерфейса с внешней шиной
TM	1	I	Режим тестирования процессора
WDT	1	O	Выход сторожевого таймера

Примечания

1 Для обозначения выводов с активным низким уровнем сигнала в качестве первого символа имени используется «X».

2 Используемые обозначения типов выводов:

I – вход,
O – выход,
O(Z) – выход с высокоимпедансным состоянием,
I/O – двунаправленный вывод.

1.3.3 Процессорная система NMPU0

Структурная схема процессорной системы NMPU0 приведена на рисунке 1.2.

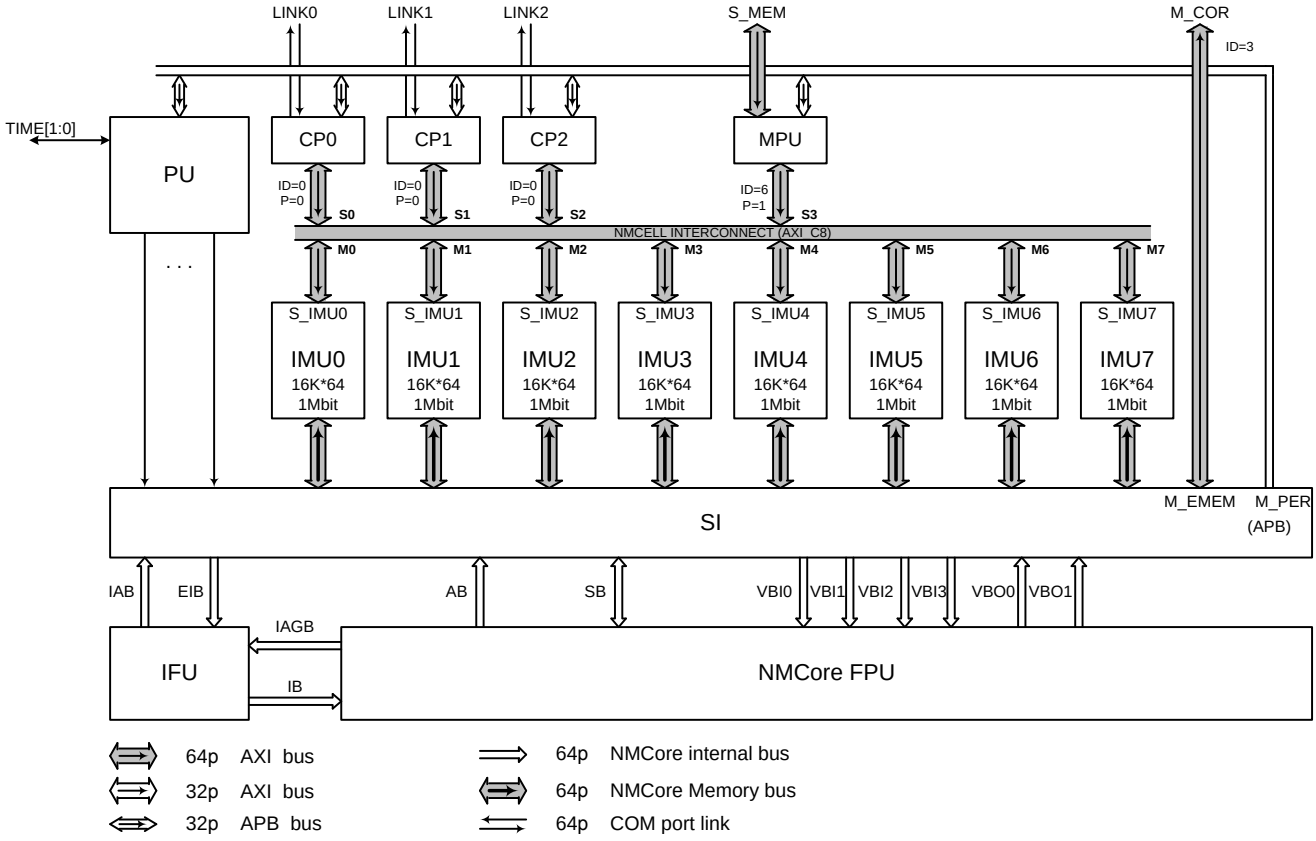


Рисунок 1.2 - Структурная схема процессорной системы NMPU0

Процессорная система NMPU0 содержит следующие функциональные узлы:

NMCore FPU – процессорное ядро NMC4, содержащее RISC-процессор и матрично-векторный сопроцессор арифметики с плавающей точкой. Обмен ядра с внешним миром осуществляется с помощью следующих 64-разрядных шин: команд (IB), скалярных данных (SB), четырёх шин векторных входных данных (VBI0 – VBI3) и двух шин векторных выходных данных (VBO0 – VB1). Кроме того, используются шина адреса перехода по команде (IAGB) и шина адреса данных (AB).

IFU (Instruction Fetch Unit) - блок предвыборки команд, который выстраивает в единую очередь команды, считываемые из внутренней или внешней памяти системы по шине EIB в соответствии с адресом, выставляемым по шине IAB.

SI (System Integrator) – системный интегратор, который осуществляет доступ во внешнюю и внутреннюю память за данными или к периферийным регистрам по запросу от процессорного ядра NMCore FPU, а также во внешнюю и внутреннюю память за командами по запросу от IFU. Соответственно, для доступа во внешнюю память используется порт M_EMEM, являющийся мастером для шины AXI (M_COR), а к периферии – порт M_PER, являющийся мастером для периферийной шины APB.

IMU0-IMU7 (Internal Memory Unit 0-7) – 8 банков внутренней памяти объёмом по 1 Мбит (16К x 64 разряда) каждый. Физически банк внутренней памяти состоит из двух полубанков памяти типа SRAM с организацией 8К x 64 каждый и имеет два порта – один со стороны процессорного ядра, другой – от каналов ПДП. Выбор полубанка памяти, в который

производится обращение, осуществляется в зависимости от значения первого разряда адреса выполняемой операции (чтения или записи). Если первые разряды адресов обращений со стороны порта процессорного ядра и со стороны порта ПДП не совпадают, то будут обслужены оба запроса, а если первые разряды совпадают, то обслуживается только один запрос в соответствии с установленным приоритетом.

MPU (Memory Protect Unit) – блок защиты памяти процессорной системы NMPU0. К нему приходит запрос на доступ во внутреннюю память системы извне по порту S_MEM, который является ведомым портом шины AXI. Если запрос попадает в разрешённую область памяти, он обслуживается обычным образом, если нет, то он блокируется с формированием соответствующего прерывания.

CP0 - CP3 (Communication Port 0 - 3) – контроллеры ПДП для коммуникационных портов 0 – 3. Данные извне для процессорной системы поступают по двунаправленным 64-разрядным линкам (LINK0 – LINK2), буферизуются и в режиме ПДП записываются в один из внутренних банков этой системы (IMU0-IMU7).

Выходные шины блоков CP0 - CP3, MPU и входные шины внутренних банков памяти IMU0-IMU7 объединяются с помощью шинного коммутатора **NMCELL INTERCONNECT**, работающего согласно стандарту AXI 3.0. Данный коммутатор имеет 4 ведомых порта (S0 – S3) и 8 ведущих (мастер) порта (M0 – M7), что позволяет одновременно удовлетворить до четырёх запросов на доступ во внутреннюю память процессорной системы.

PU (Peripheral Unit) – блок периферийных регистров процессорного ядра: регистры запросов и масок внешних прерываний, регистры управления и состояния двух интервальных таймеров, регистры управления и состояния контроллеров ПДП для коммуникационных портов 0 – 3.

1.3.4 Процессорная система NMPU1

Структурная схема процессорной системы NMPU1 приведена на Рисунок 1.3. Данная процессорная система аналогична системе NMPU0 за исключением:

- процессорное ядро в NMPU1 использует матрично-векторный сопроцессор для обработки данных, представленных в формате с фиксированной точкой и программируемой разрядностью, и не поддерживает арифметику с плавающей точкой;
- процессорное ядро в NMPU1 требует максимум четыре одновременных доступа в память за векторными данными, поэтому число внутренних банков процессорной системы уменьшено до четырёх;

Процессорная система NMPU1 содержит следующие функциональные узлы:

NMCore INT – процессорное ядро NMC4, содержащее RISC-процессор и матрично-векторный сопроцессор для обработки данных, представленных в формате с фиксированной точкой и программируемой разрядностью. Обмен ядра с внешним миром осуществляется с помощью следующих 64-разрядных шин: команд (IB), скалярных данных (SDB), векторных входных данных (VDIB), весов (WB), векторного регистра (VRB) и векторных выходных данных (VDOB). Кроме того, используются шина адреса перехода по команде (IAGB) и шина адреса данных (AB).

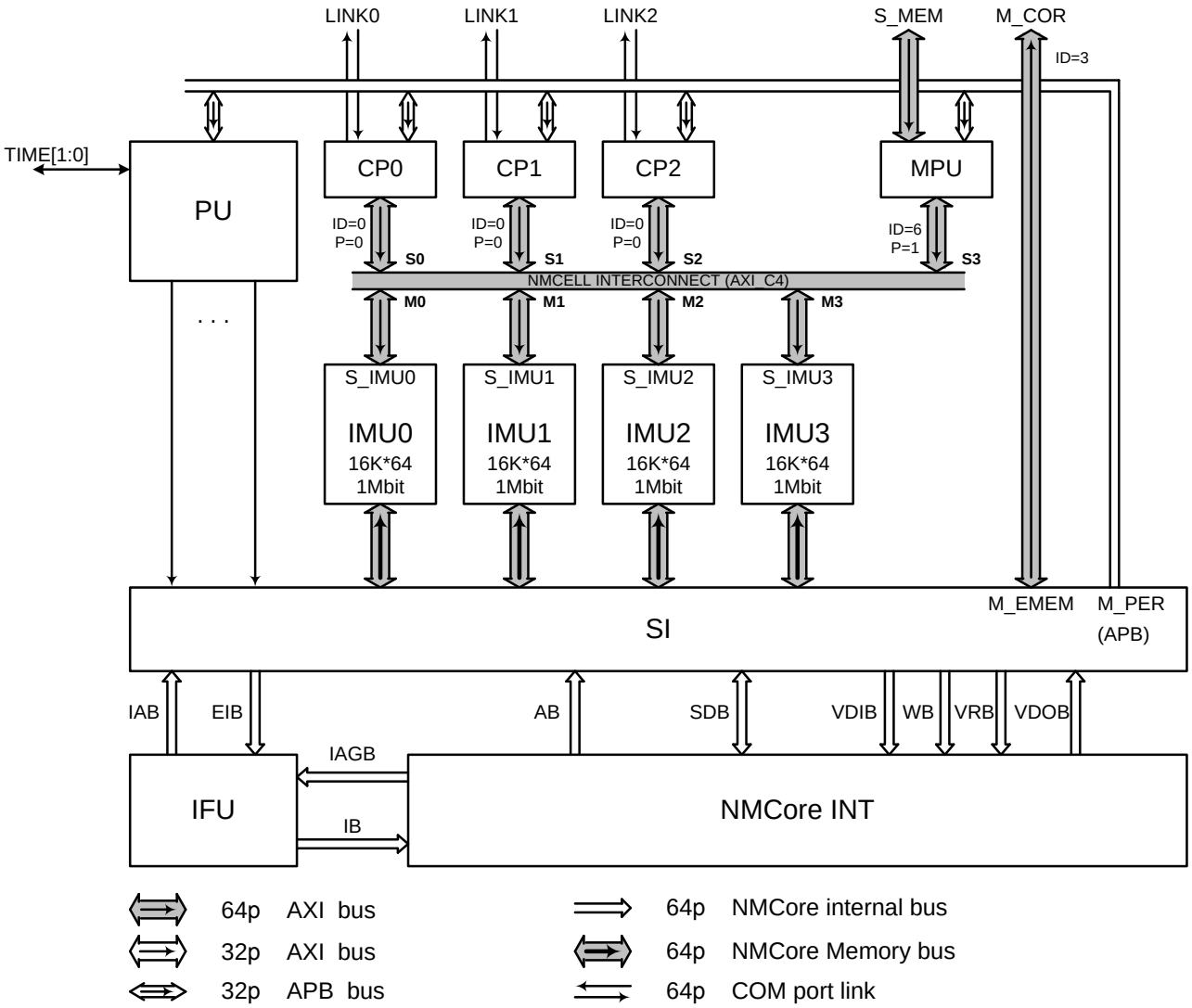


Рисунок 1.3 - Структурная схема процессорной системы NMPU1

IFU (Instruction Fetch Unit) - блок предвыборки команд, который выстраивает в единую очередь команды, считываемые из внутренней или внешней памяти системы по шине EIB в соответствии с адресом, выставляемым по шине IAB.

SI (System Integrator) – системный интегратор, который осуществляет доступ во внешнюю и внутреннюю память за данными или к периферийным регистрам по запросу от процессорного ядра NMCore INT, а также во внешнюю и внутреннюю память за командами по запросу от IFU. Соответственно, для доступа во внешнюю память используется порт M_EMEM, являющийся мастером для шины AXI (M_COR), а к периферии – порт M_PER, являющийся мастером для периферийной шины APB.

IMU0-IMU3 (Internal Memory Unit 0-3) – 4 банка внутренней памяти объёмом по 1 Мбит (16К х 64 разряда) каждый. Физически банк внутренней памяти состоит из двух полубанков памяти типа SRAM с организацией 8К х 64 каждый и имеет два порта – один со стороны процессорного ядра, другой – от каналов ПДП. Выбор полубанка памяти, в который производится обращение, осуществляется в зависимости от значения первого разряда адреса выполняемой операции (чтения или записи). Если первые разряды адресов обращений со стороны порта процессорного ядра и со стороны порта ПДП не совпадают, то будут

обслужены оба запроса, а если первые разряды совпадают, то обслуживается только один запрос в соответствии с установленным приоритетом.

MPU (Memory Protect Unit) – блок защиты памяти процессорной системы NMPU1. К нему приходит запрос на доступ во внутреннюю память системы извне по порту S_MEM, который является ведомым портом шины AXI. Если запрос попадает в разрешённую область памяти, он обслуживается обычным образом, если нет, то он блокируется с формированием соответствующего прерывания.

PU (Peripheral Unit) – блок периферийных регистров процессорного ядра.

CP0 - CP3 (Communication Port 0 - 3) – контроллеры ПДП для коммуникационных портов 0 – 3. Данные извне для процессорной системы поступают по двунаправленным 64-разрядным линкам (LINK0 – LINK2), буферизуются и в режиме ПДП записываются в один из внутренних банков этой системы **(**IMU0-IMU3).

Выходные шины блоков CP0 - CP3, MPU и входные шины внутренних банков памяти IMU0-IMU7 объединяются с помощью шинного коммутатора **NMCELL INTERCONNECT**, работающего согласно стандарту AXI 3.0. Данный коммутатор имеет 4 ведомых порта (S0 – S3) и 4 ведущих (мастер) порта (M0 – M3), что позволяет одновременно удовлетворить до четырёх запросов на доступ во внутреннюю память процессорной системы.

1.3.5 Карта памяти

Карта памяти микросхемы 1879ВМ6Я приведена на Рисунок 1.4.

Поскольку микросхема содержит две процессорных системы NMPU0 и NMPU1, на рисунке приведена карта памяти для каждой такой системы. Кроме того, добавлена карта памяти для контроллеров ПДП: MDMAC, KDMAC и USB, поскольку они могут обмениваться данными с памятью независимо от процессорных систем. Каждая процессорная система может адресовать до 4 Гбайт, но до 32-разрядного слова. В отличие от них контроллеры ПДП имеют доступ в память до байта. Карты памяти всех процессорных систем похожи. По младшим адресам расположены собственные банки внутренней памяти, где хранятся программы обработки прерываний и начальный загрузчик. Затем идут общие банки внутренней памяти. В старших адресах карты памяти расположены общие периферийные устройства: USB, MDMAC, контроллер EMI (адреса для шины AXI), затем SC, DIT, WDT, GPIO, SPI, ETIRC. Потом идёт общий банк внешней памяти EMI, и заканчивается карта памяти адресами периферии, своей для каждой процессорной системы. Таким образом, каждая процессорная система может адресоваться к своим банкам, общим и чужим банкам внутренней памяти, к общим периферийным устройствам, к общей внешней памяти и к своим периферийным устройствам.

Карта памяти для контроллеров ПДП является отражением карты памяти процессорных систем, откуда исключены свои банки внутренней памяти и своя периферия.

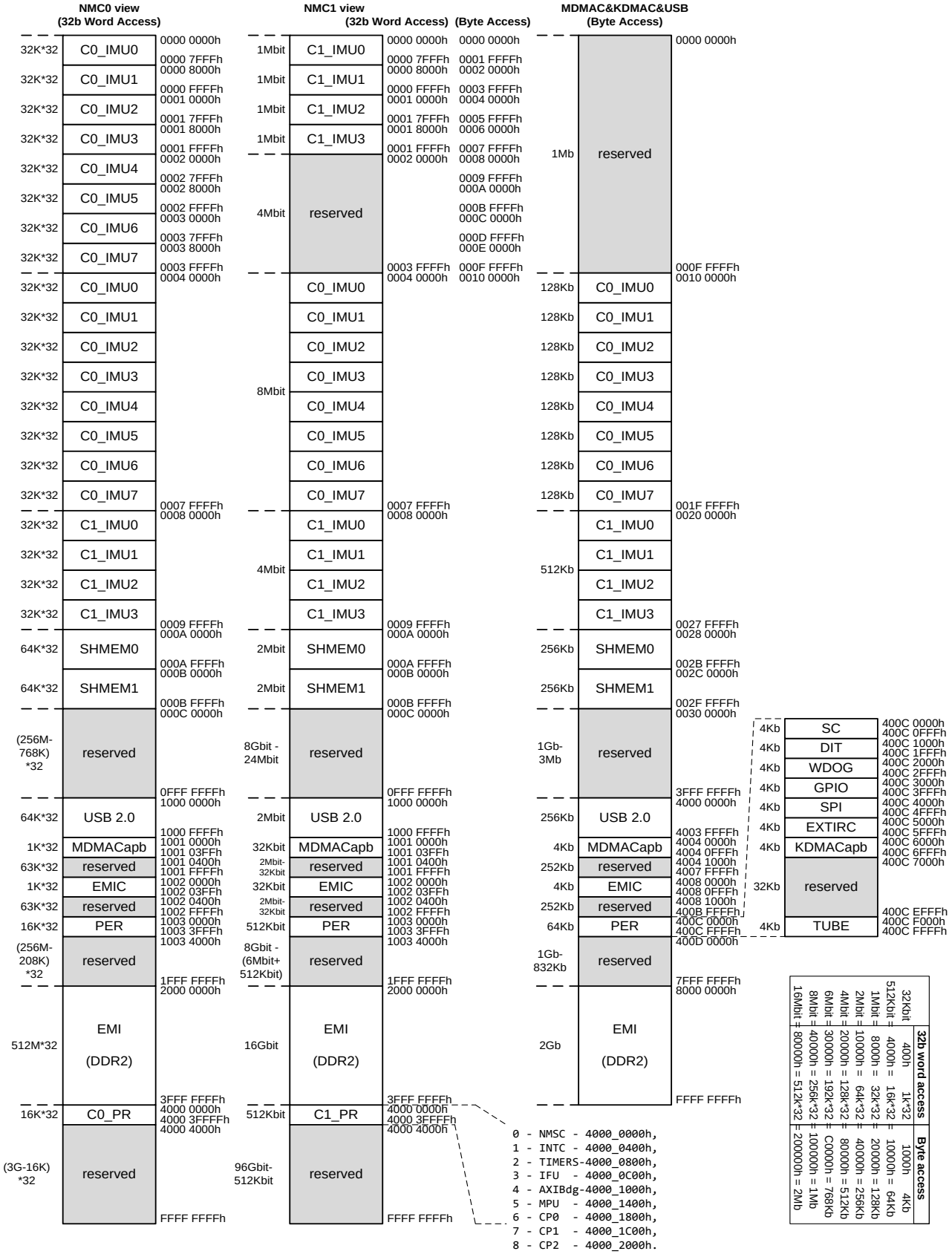


Рисунок 1.4 – Карта памяти микросхемы

1.4 Архитектурные особенности микросхемы

1.4.1 Обобщённая структура процессорной системы цифровой обработки сигналов на базе процессорного ядра NMC4

Требование обработки потоков данных в реальном времени с производительностью, сравнимой с производительностью универсальных процессоров, и потреблением, характерным для встроенных систем, привело к созданию в ЗАО НТЦ «Модуль» архитектуры NeuroMatrix, обладающей следующими основными особенностями:

- Векторно-конвейерный принцип выполнения операций, позволяющий одним потоком команд задавать большое количество параллельно выполняемых операций (динамический VLIW);
- Наличие одного или нескольких векторных сопроцессоров, работающих под управлением одного RISC-процессора и имеющих свои шины ввода/вывода данных;
- Использование внешних адресных генераторов, что обеспечивает эффективное использование адресных регистров ядра при адресации векторных данных;
- Конвейер с очередью команд на ступени выборки операндов из памяти, обеспечивающий эффективную работу с банками внутренней и внешней памяти, имеющими различную глубину конвейера;
- Выдача и приём данных на одной и той же ступени конвейера (Late Write), что резко снижает количество конфликтов по данным;
- Использование команд, задающих несколько операций (статический VLIW).

Обобщенная структура процессорной системы цифровой обработки сигналов на базе процессорного ядра NMC4 (NeuroMatrix Core 4), обладающая всеми вышеперечисленными свойствами, приведена на Рисунок 1.5).

В состав процессорной системы цифровой обработки сигналов входят следующие основные блоки:

NMC4 – процессорное ядро, содержащее управляющий RISC-процессор и несколько сопроцессоров, выполняющих цифровую обработку векторных данных.

RISC-процессор решает следующие основные задачи:

- Декодирует и выполняет команды, считываемые из памяти по шине команд (Instruction Bus);
- Осуществляет через шину скалярных данных (Scalar Data Bus) конфигурирование сопроцессоров, настраивая их на обработку данных;
- Выставляет на внутреннюю шину инструкций (Internal Instruction Bus) команды оперативного управления сопроцессорами;
- Формирует и выставляет на адресную шину (Address Bus) адреса команд, а также скалярных и векторных данных.

Internal Memory Bank – банки внутренней памяти.

External Memory Interface – интерфейсы с внешней памятью.

Instruction Address Generator – генератор последовательных адресов команд, содержимое которого может быть изменено произвольным образом при выполнении различных команд перехода.

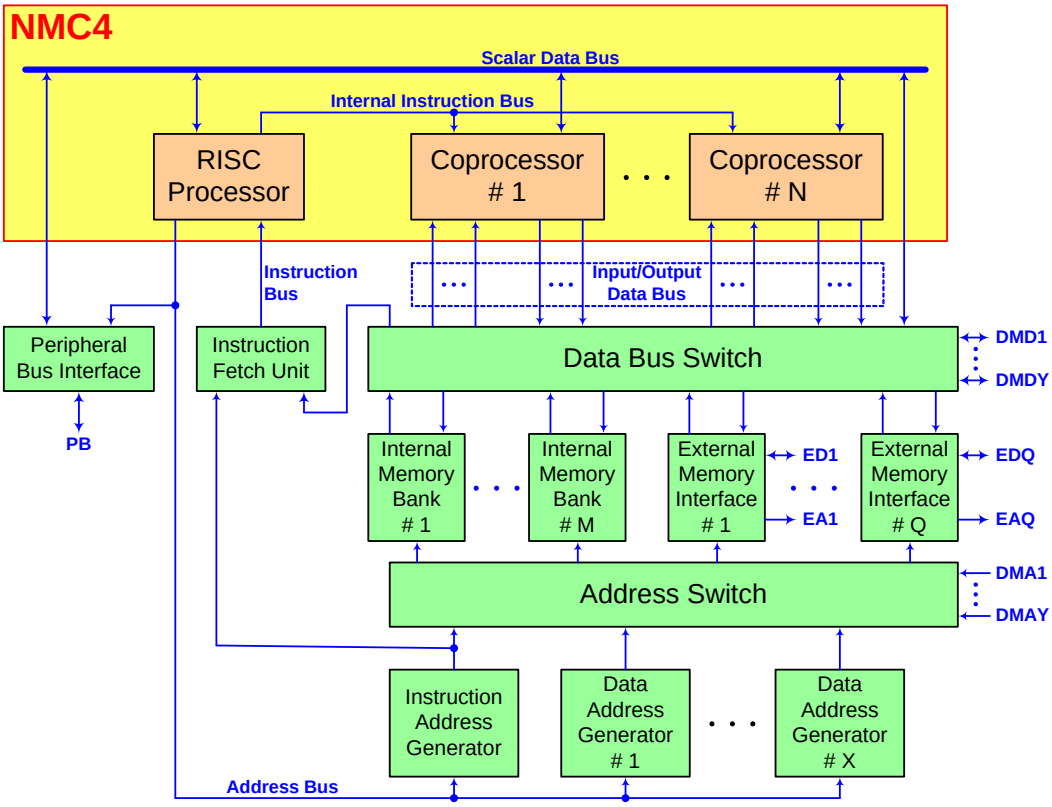


Рисунок 1.5 - Обобщенная структура процессорной системы цифровой обработки сигналов на базе процессорного ядра NeuroMatrix 4

Data Address Generator – генераторы адресов векторных данных, загружаемые через адресную шину начальным адресом, смещением адреса и количеством повторов операции ввода/вывода.

Address Switch – коммутатор адресов, который осуществляет пересылку адресов, формируемых адресными генераторами, и адресов, поступающих от внешних каналов прямого доступа к памяти, в банки внутренней памяти и в интерфейсы с внешней памятью.

Data Bus Switch – коммутатор шин данных, обеспечивающий обмен данными между процессорным ядром и памятью системы, а также обмен данными между каналами DMA и банками памяти.

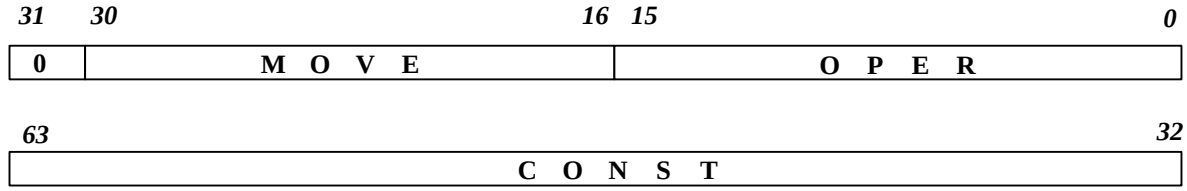
Instruction Fetch Unit – блок выборки команд, который выстраивает в единую очередь команды, считываемые из внутренней или внешней памяти системы.

Peripheral Bus Interface – интерфейс с шиной периферийных устройств, через которую процессор осуществляет инициализацию, настройку и оперативное управление внешними устройствами, а также обмен данными с различными портами ввода/вывода информации.

Количество и тип сопроцессоров, количество банков внутренней памяти и внешних шин системы, а также количество генераторов адресов векторных данных определяются требуемой производительностью системы, что позволяет легко наращивать производительность вычислительных систем в зависимости от класса решаемых задач с сохранением программной совместимости внутри семейства процессоров.

1.4.2 Статический VLIW

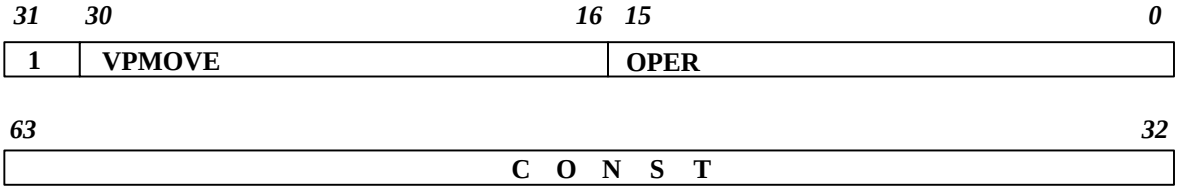
Команды процессорного ядра NMC4 делятся на две основные группы: команды управляющего RISC-процессора (см. Рисунок 1.6) и команды сопроцессора арифметики с плавающей точкой (см. Рисунок 1.7). Каждая из этих групп, в свою очередь, делится на команды скалярные, т. е. обычные RISC-команды, и векторные, которые задают многократно одни и те же действия над векторами данных, что эквивалентно аппаратной поддержке коротких циклов.



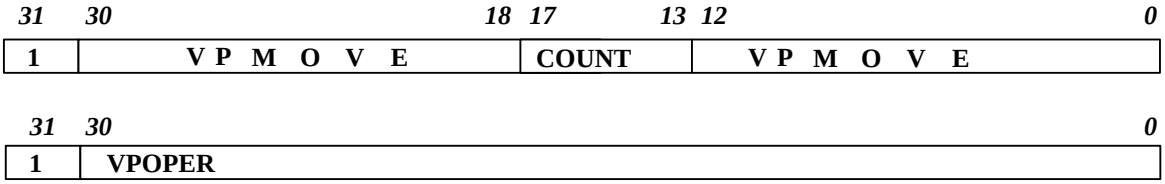
а) Кодировка скалярных команд управляющего RISC-процессора

б) Кодировка векторных команд управляющего RISC-процессора

Рисунок 1.6 - Кодировка команд управляющего RISC-процессора



а) Кодировка скалярных команд сопроцессора арифметики с плавающей точкой



б) Кодировка векторных команд сопроцессора арифметики с плавающей точкой

Рисунок 1.7 - Кодировка команд сопроцессора арифметики с плавающей точкой

Скалярные команды управляющего RISC-процессора содержат следующие поля: MOVE – задаёт ввод/вывод данных с одновременной модификацией адресных регистров, условные переходы/переходы к подпрограмме и возвраты из подпрограммы/прерывания; OPER – определяет арифметическую, логическую операцию или операцию сдвига. Эти команды могут использовать 32-разрядные константы (поле CONST), которые могут грузиться в регистры или использоваться для задания адреса или смещения при обращении к памяти.

Векторные команды управляющего RISC-процессора задают операции над данными, представленными в формате с фиксированной точкой, и имеют похожие поля: VMOVE – задаёт ввод/вывод векторных данных, VOPER – определяет арифметическую или логическую операцию над векторными данными, COUNT – определяет число повторов выполнения

данной команды (от 1 до 32), что позволяет аппаратно поддержать организацию коротких циклов и значительно увеличить плотность кода.

Команды сопроцессора арифметики с плавающей точкой имеют следующие отличия от команд управляющего RISC-процессора:

- Нельзя задать команды управления (переходы/переходы к подпрограмме и возвраты из подпрограммы/прерывания);
- Ввод/вывод данных осуществляется только в регистры сопроцессора арифметики с плавающей точкой в соответствии с полем VPMOVE, но адресация в память осуществляется по-прежнему управляющим RISC-процессором;
- Векторные арифметические операции в формате с плавающей точкой задаются полем VPOPER, причём данные арифметические операции не совмещаются с вводом/выводом векторных данных.

В остальном скалярные и векторные команды сопроцессора арифметики с плавающей точкой и соответствующие команды управляющего RISC-процессора задают аналогичные операции.

Таким образом, процессор использует команды типа VLIW, задающие одновременно операции обмена с памятью, модификацию адресных регистров и арифметическую операцию, причём это относится как к скалярным, так и к векторным командам обработки данных в формате с фиксированной точкой. Объединение в одной команде операций ввода-вывода и арифметической операции позволяет увеличить производительность скалярных команд на реальных задачах до 40 %, но для векторных команд это решение, по нашим оценкам, особый выигрыш не даёт. Поэтому для команд сопроцессора арифметики с плавающей точкой такого типа совмещения нет.

1.4.3 Многотактовые векторные команды и векторно- конвейерная организация вычислений (динамический VLIW)

Принцип организации выполнения одновременно нескольких операций, заданных разными векторными командами, в разных функциональных узлах одного или нескольких сопроцессоров показан на Рисунок 1.8. Несмотря на то, что команды поступают на выполнение по одной и в строгой последовательности, за счёт использования многотактовых команд достигается эффект суперскаляра. Также поддерживается внеочередное выполнение команд (out-of-order execution).

Рисунок 1.8 - Векторно-конвейерная организация вычислений (динамический VLIW)

1.4.4 Особенности работы конвейера команд при обмене данных с памятью

Принципы организации конвейера команд процессора цифровой обработки сигналов на базе процессорного ядра NeuroMatrix 4 показан на Рисунок 1.9.

Его основными особенностями являются:

- Наличие общих первой и второй ступеней для всех команд, причем на первой ступени осуществляется вычисление адреса первого данного для команды, последнего адреса и смещения (для векторных команд), модификации адресных регистров, на второй организуется единая очередь команд, ожидающих своих данных перед выполнением;
- Несколько параллельных подконвейеров на третьей ступени (стадии выполнения операций), причём ввод и вывод данных осуществляется именно на данной ступени (реализуется Late Write).

Данный конвейер позволяет обеспечить эффективную работу с банками внутренней и внешней памяти, имеющими различную глубину конвейера без потери производительности. Реализация выдачи и приёма данных на одной и той же ступени конвейера (Late Write) резко снижает количество конфликтов по данным.

Наличие очереди команд, ожидающих своих данных, позволяет обеспечить эффективную работу с банками внутренней и внешней памяти, имеющими различную глубину конвейера без потери производительности. Данная очередь имеет глубину восемь, что позволяет эффективно работать с внешней синхронной памятью в конвейерном режиме - имеется возможность выставить до восьми запросов на чтение, прежде чем придут первые данные.

Если процессор работает только с внутренней памятью, очередь команд заполнена не полностью. Стоит произвести хоть одно обращение во внешнюю память, команда, ожидающая своих данных из внешней памяти, не может уйти из очереди, и очередь начинает заполняться. Как только данные придут, очередь начинает разгружаться. Тем самым реализуется конвейер переменной глубины в зависимости от требуемого числа тактов обращения в память, и это даёт возможность эффективно работать одновременно как с внешней, так и внутренней памятью.

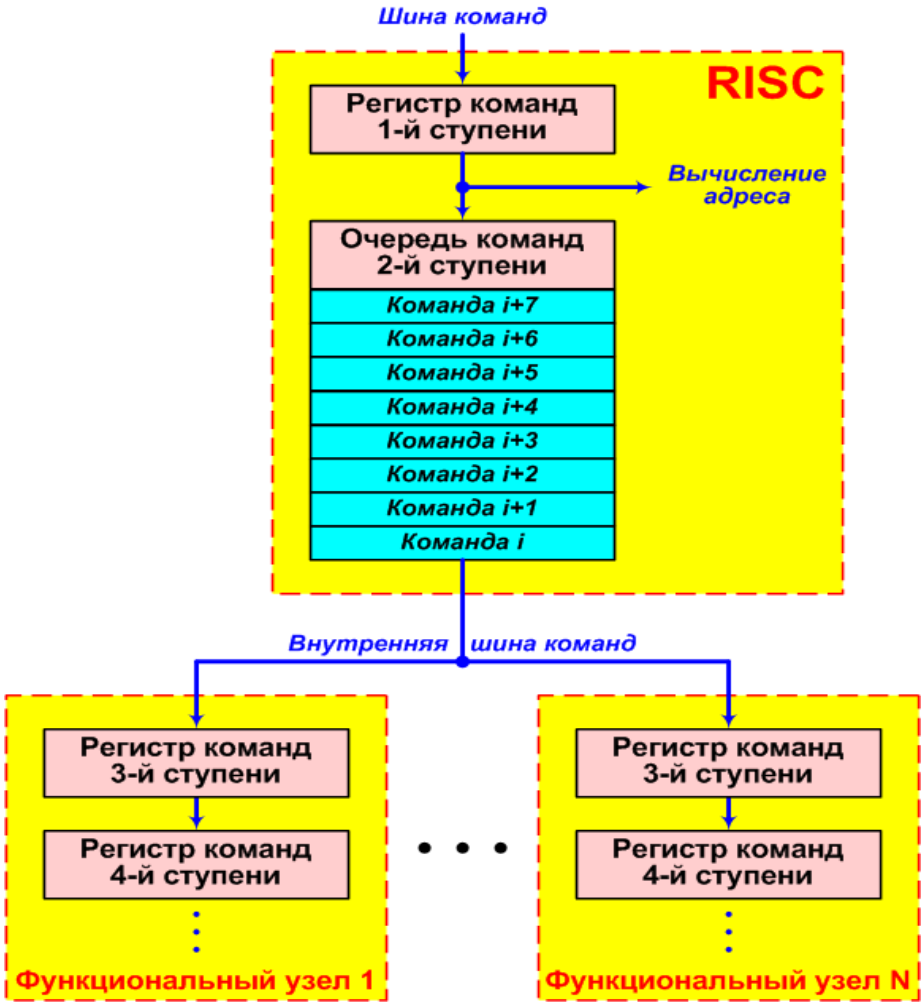


Рисунок 1.9 - Конвейер команд процессора цифровой обработки сигналов на базе процессорного ядра NeuroMatrix 4

1.4.5 Единый адресный генератор процессорного ядра

Как было описано ранее, все команды при запуске на выполнение, как и раньше, начинают свою работу на общей части первого уровня конвейера. Скалярная команда на данной ступени конвейера выставляет запрос на доступ в память. В случае отсутствия блокировок по доступу в память и от нижних ступеней конвейера команда попадает на вторую ступень конвейера и освобождает общую часть.

Векторная команда при запуске занимает общую часть первого уровня конвейера на один процессорный такт. За это время она вычисляет адрес первого обращения в память. Одновременно на специальном арифметическом устройстве, содержащем умножитель 5*32 разряда и 32-разрядный сумматор, формируется новое значение адресного регистра, использующегося в качестве базы. Это значение совпадает с тем, что должно получиться после завершения выполнения векторной команды. В следующем такте векторная команда освобождает первый уровень конвейера, уходит на второй уровень и одновременно занимает один из адресных генераторов, находящихся вне процессорного ядра. При этом запоминается адрес первого обращения в память, сохраняется копия значения регистра- смещения и формируется запрос на доступ в память. Далее внешний адресный генератор работает самостоятельно, не занимая ресурсов процессорного ядра.

Наличие единого адресного генератора у процессорного ядра позволяет использовать один и тот же адресный регистр этого ядра в качестве источника адреса для нескольких команд, при этом следующая команда не должна ждать полного окончания предыдущей. При этом блок адресных генераторов вне процессорного ядра обеспечивает генерацию запросов на обмен с памятью для нескольких скалярных и векторных команд одновременно.

1.4.6 Аппаратная вершина системного стека

В микросхеме введена аппаратная вершина системного стека. Она копирует содержимое ячейки памяти системного стека, хранящей последний адрес возврата из подпрограммы или прерывания и значение регистра слова состояния процессора PSWR при входе в подпрограмму/прерывание. Поддержка аппаратной вершины системного стека позволяет резко ускорить выполнение команд возврата из подпрограммы или прерывания, особенно если системный стек расположен во внешней по отношению к ядру памяти.

1.5 Система прерываний

1.5.1 Типы прерываний

Процессорные системы NMPU0 и NMPU1 имеют развитую систему прерываний, позволяющую оперативно реагировать на внешние и внутренние события, а также управлять различными периферийными устройствами. Прерывания процессорных систем представлены в Таблица 1.2 в порядке уменьшения приоритета сверху вниз. Исключение составляет пошаговое прерывание, которое имеет наименьший приоритет. Для процессорной системы NMPU1 прерывания с номерами 0-5 всегда равны нулю, поскольку они формируются сопроцессором арифметики с плавающей точкой (данный сопроцессор имеется только в системе NMPU0).

Таблица 1.2 - Прерывания процессорных систем NMPU0 и NMPU1

№	Прерывание	Обозначение	Адрес вектор
1	Немаскируемое прерывание	NMI	0x0000_0000 hex
2	Переполнение	OF	0x0000_0008 hex
3	Неправильная команда	EI	0x0000_0010 hex
4	Пошаговое прерывание	ST	0x0000_0018 hex
5	Прерывание 0 от сопроцессора арифметики с плавающей точкой (неправильная команда)	FPIEI	0x0000_0020 hex-
6	Прерывание 1 от сопроцессора арифметики с плавающей точкой (некорректные данные)	FPIOI	0x0000_0028 hex-
7	Прерывание 2 от сопроцессора арифметики с плавающей точкой (переполнение)	FPOI	0x0000_0030 hex-
8	Прерывание 3 от сопроцессора арифметики с плавающей точкой (потеря значимости)	FPUI	0x0000_0038 hex-
9	Прерывание 4 от сопроцессора арифметики с плавающей точкой (потеря точности)	FPII	0x0000_0040 hex-
10	Прерывание 5 от сопроцессора арифметики с плавающей точкой (потеря данного)	FPDLI	0x0000_0048 hex
11	Прерывание от системного интегратора (обращение блока выборки команд в периферийную область)	IFE	0x0000_0050 hex
12	Прерывание от моста «системный интегратор – AXI» (ошибка обращения во внешнюю память)	DAE	0x0000_0058 hex
13	Прерывание от блока защиты памяти (защита по записи)	WMP	0x0000_0060 hex
14	Прерывание от блока защиты памяти (защита по чтению)	RMP	0x0000_0068 hex
15	Прерывание от блока таймеров (таймер 0)	T0	0x0000_0070 hex
16	Прерывание от блока таймеров (таймер 1)	T1	0x0000_0078 hex
17	Прерывание 0 от системного контроллера (межпроцессорное прерывание)	PI0	0x0000_0080 hex
18	Прерывание 1 от системного контроллера (межпроцессорное прерывание)	PI1	0x0000_0088 hex
19	Прерывание 2 от системного контроллера (межпроцессорное прерывание)	PI2	0x0000_0090 hex
20	Прерывание 3 от системного контроллера (межпроцессорное прерывание)	PI3	0x0000_0098 hex
21	Прерывание от блока сторожевого таймера WDT	WDT	0x0000_00A0 hex
22	Прерывание 0 от блока интервальных таймеров (DIT)	DIT0	0x0000_00A8 hex
23	Внешнее прерывание 0	INT0	0x0000_00B0 hex
24	Внешнее прерывание 1	INT1	0x0000_00B8 hex
25	Прерывание от контроллера USB	USB	0x0000_00C0 hex
26	Прерывание от контроллера SPI	SPI	0x0000_00C8 hex
27	Прерывание от контроллера внешней памяти (EMIC)	EMIC	0x0000_00D0 hex
28	Прерывание 1 от блока интервальных таймеров (DIT)	DIT1	0x0000_00D8 hex

Продолжение таблицы 1.2

№	Прерывание	Обозначение	Адрес вектор
29	Прерывание от коммуникационного порта CP0 (передающий канал)	CPO0	0x0000_00E0 hex
30	Прерывание от коммуникационного порта CP0 (принимающий канал)	CPI0	0x0000_00E8 hex
31	Прерывание от коммуникационного порта CP1 (передающий канал)	CPO1	0x0000_00F0 hex
32	Прерывание от коммуникационного порта CP1 (принимающий канал)	CPI1	0x0000_00F8 hex
33	Прерывание от коммуникационного порта CP2 (передающий канал)	CPO2	0x0000_0100 hex
34	Прерывание от коммуникационного порта CP2 (принимающий канал)	CPI2	0x0000_0108 hex
35	Внешнее прерывание 2	INT2	0x0000_0110 hex
36	Внешнее прерывание 3	INT3	0x0000_0118 hex
37	Прерывание по завершении работы канала ПДП память-память (от MDMAC)	MDMA	0x0000_0120 hex
38	Прерывание по ошибке доступа в память канала ПДП память-память (от MDMAC)	EMDMA	0x0000_0128 hex
39	Прерывание по завершениии работы канала ПДП память-периферия (от KDMAC)	KDMA	0x0000_0130 hex
40	Прерывание по завершении работы канала ПДП периферия-память (от KDMAC)	EKDMA	0x0000_0138 hex
41	Прерывание 4 от системного контроллера (межпроцессорное прерывание)	PI4	0x0000_0140 hex
42	Прерывание 5 от системного контроллера (межпроцессорное прерывание)	PI5	0x0000_0148 hex
43	Прерывание 6 от системного контроллера (межпроцессорное прерывание)	PI6	0x0000_0150 hex
44	Прерывание 7 от системного контроллера (межпроцессорное прерывание)	PI7	0x0000_0158 hex

1.5.2 Внутренние и внешние прерывания процессорного ядра

Процессор поддерживает 44 прерывания (см. таблицу 1.2): 4 внутренних прерывания процессорного ядра (в таблице с номерами от 1 до 4), 6 прерываний от сопроцессора с плавающей точкой и 34 внешних по отношению к процессорному ядру. Такое деление прерываний на две группы объясняется реализацией двухуровневой системы прерываний, как показано на рисункеРисунок 1.10. Запросы от четырёх внешних прерываний ($\overline{\text{INT0}}$ … $\overline{\text{INT3}}$) через блок формирования внешних прерываний (EXINC), запросы от сопроцессора с плавающей точкой (FPU_COPR) и запросы от периферийных узлов (PERIPHERAL UNITS) поступают на контроллер внешних прерываний (INTC) процессорного ядра и фиксируются в специальном программно доступном периферийном регистре. Далее запросы перемножаются на программно настраиваемую маску и проходят арбитраж. Затем в соответствии с победившим запросом выставляется адрес-вектор соответствующего прерывания (External Interrupt Vector) и формируется обобщённый запрос на внешнее прерывание (External Interrupt Request). Данный запрос и адрес-вектор фиксируются в процессорном ядре NMC4, после чего сбрасывается победивший запрос на прерывание в контроллере прерываний, и данный контроллер может выставить ещё один обобщённый запрос и соответствующий адрес-вектор прерывания. Следующее внешнее прерывание будет зафиксировано процессорным ядром только после того, как будет обработано предыдущее внешнее прерывание. Более подробно о работе контроллера внешних прерываний см. в разделе 7.10.

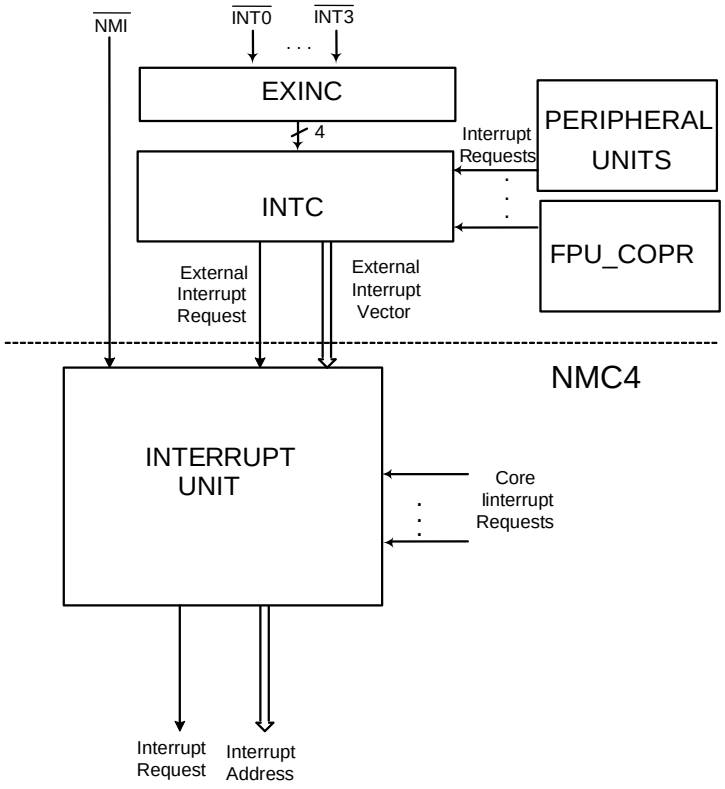


Рисунок 1.10 - Структурная схема системы прерываний

В процессорном ядре имеется блок прерываний (INTERRUPT UNIT), куда поступают немаскируемое прерывание $\overline{\text{NMI}}$, 4 прерывания от ядра (Core Interrupt Requests) и, как описано выше, адрес-вектор внешнего по отношению к ядру прерывания (External Interrupt Vector) и обобщённый запрос на внешнее прерывание (External Interrupt Request). Все эти запросы на прерывание, а также адрес-вектор внешнего прерывания, фиксируются в специальном программно доступном на чтение и сброс регистре INTR. Далее запросы перемножаются на программно настраиваемую маску (кроме немаскируемого прерывания) и проходят арбитраж. Причём обобщённый запрос на внешнее прерывание также имеет свою маску. Затем в соответствии с победившим запросом выставляется адрес перехода на программу обработки соответствующего прерывания (Interrupt Address) и формируется окончательный запрос на прерывание (Interrupt Request). После того, как будет разрешена обработка запроса победившего прерывания, соответствующий запрос на прерывание в ядре будет сброшен. Разрешение на обработку следующего прерывания будет получено лишь после того, как закончатся все контекстные переключения при входе в обработку предыдущего прерывания.

2 RISC- ядро

2.1 Структура RISC- ядра

RISC-ядро является одним из основных узлов процессорного ядра. Оно обеспечивает выборку и дешифрацию команд, хранение и модификацию программного счетчика PC, управление всеми исполнительными конвейерами процессора и их синхронизацию, необходимые предварительные адресные вычисления и передачу адресов к внешним адресным генераторам, а также выполняет арифметические и логические операции над 32-разрядными данными в дополнительном коде, когда использование векторного узла неэффективно. Структурная схема RISC- ядра представлена на Рисунок 2.1.

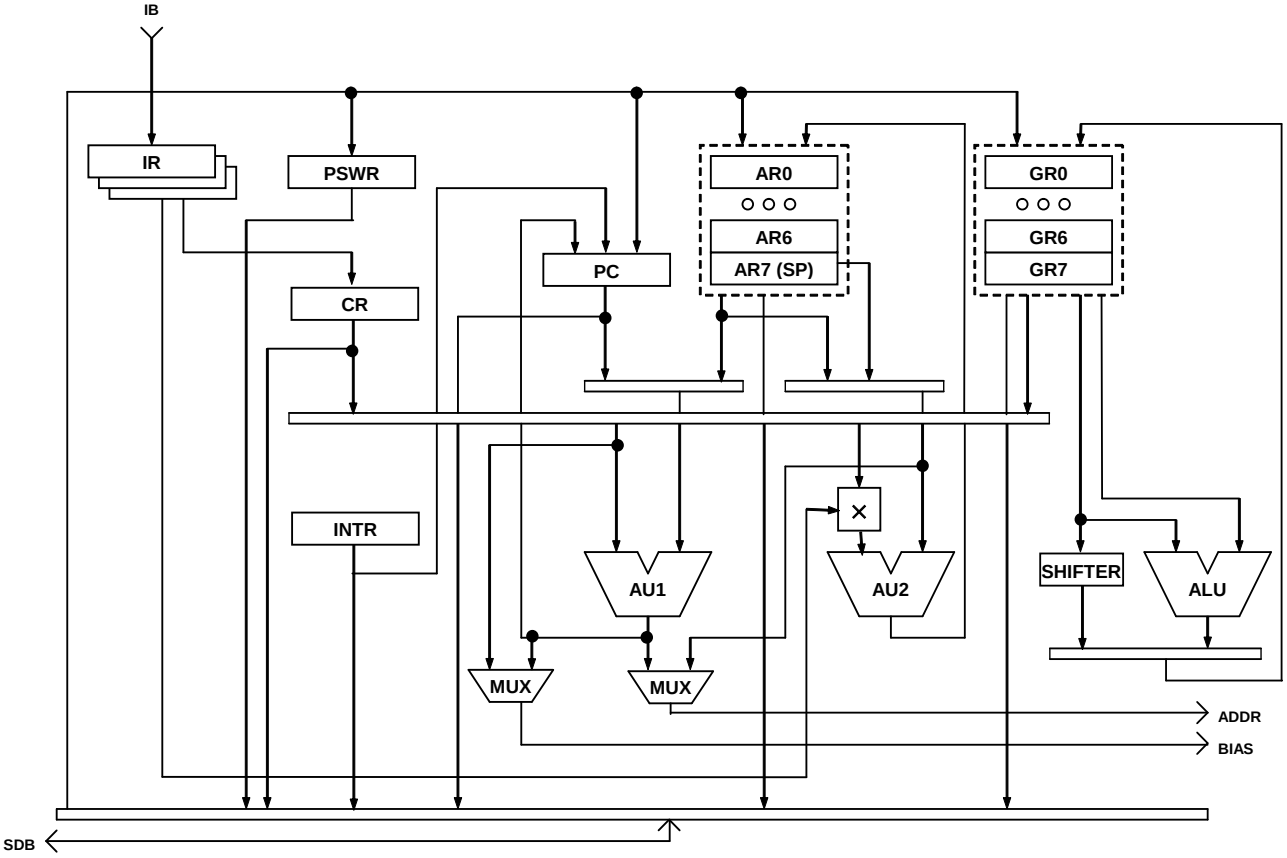


Рисунок 2.1 - Структурная схема RISC- ядра

Ниже приводится описание основных составляющих RISC- ядра:

GR0- GR7 – восемь 32-разрядных регистров общего назначения, которые используются для задания смещения при вычислении адресов команд и данных, а также в качестве источников и приёмников скалярных арифметических операций.

AR0- AR7 - восемь 32-разрядных регистров адреса, которые необходимы при вычислении адресов команд и данных. Адресный регистр AR7 (SP) является также указателем системного стека.

ALU - арифметико-логическое устройство, которое выполняет логические и арифметические операции, пошаговые операции 32-разрядного умножения над регистрами общего назначения. Результат операций также записывается в регистры общего назначения.

SHIFTER – сдвигатель, который выполняет операции сдвига над регистрами общего назначения. Результат операций также записывается в один из регистров общего назначения.

AU1 – первое адресное устройство, которое используется для формирования адреса чтения из памяти/записи в память скалярных данных или первых векторных данных, а также для вычисления адреса перехода в командах условного перехода и перехода к подпрограммам. Оно использует в качестве операндов значение адресного регистра либо счетчика команд PC в качестве адреса и значение регистра общего назначения, либо непосредственную константу из команды в качестве смещения. Результат вычислений может быть выдан на внешнюю шину адреса или смещения адреса, а также быть записан в программном счетчике PC.

AU2 – второе адресное устройство, которое используется для модификации адресных регистров при выполнении как скалярных, так и векторных команд. Оно берёт в качестве операндов значение адресного регистра либо указателя стека как базу и значение регистра общего назначения как смещение. В случае выполнения векторной команды смещение перед адресными вычислениями увеличивается в число раз, соответствующее числу повторений соответствующей векторной команды. Результат операции сохраняется в адресном регистре.

PC - счетчик команд, показывающий на адрес следующей команды, которая поступит на выполнение.

CR – буфер констант. Он хранит константу, задаваемую командой, для её использования при адресных вычислениях или для её записи в программно доступные регистры как процессорного ядра, так и периферийных узлов.

PSWR – регистр слова состояния процессора, который содержит маски на прерывания, флаги (признаки) выполнения арифметических команд, номер текущего регистрового окна, информацию для аппаратной поддержки вершины системного стека.

IR - конвейерные, программно недоступные регистры команд. Они обеспечивают управление всеми конвейерами RISC-ядра и матрично-векторного сопроцессора.

2.2 Основные режимы работы RISC- ядра и методы адресации памяти

Структура RISC-процессорного ядра выбрана таким образом, чтобы он мог работать одновременно в двух режимах:

- Поддержка операций матрично-векторного сопроцессора, когда ядро вычисляет адрес следующей команды, а также адрес первых векторных данных и смещение, необходимых сопроцессору;
- Поддержка команд управления (организация циклов, ветвлений и т. д.) и скалярных команд обработки данных, когда RISC-ядро вычисляет адрес следующей команды, исполнительный адрес для скалярных операндов, а также выполняет арифметические и логические операции над данными в GR0- GR7.

Адрес следующей команды может быть определен либо с помощью инкрементации PC, либо путем его задания константой в коде команды, либо сложением содержимого одного из адресных регистров - AR0, …, AR7 или PC со смещением, заданным константой в коде команды или содержимым одного из регистров общего назначения - GR0, ..., GR7. Тем самым, в RISC- процессорном ядре поддерживаются следующие команды управления (как условные, так и безусловные): переход/переход со смещением (JUMP/SKIP), переход к подпрограмме (CALL), возврат из подпрограммы/прерывания (RET/RETI), останов (HALT).

Исполнительный адрес операндов может вычисляться с использованием следующих методов адресации: по содержимому адресного регистра без его изменения, с его инкрементацией, декрементацией, а также по сумме содержимого адресного регистра и соответствующего регистра общего назначения. Для скалярных команд кроме этого возможно определение исполнительного адреса с помощью константы, задаваемой в поле команды.

2.3 Регистр слова состояния процессора PSWR

32-разрядный регистр слова состояния процессора PSWR содержит всю наиболее важную информацию о работе процессорного ядра. Данный регистр сохраняется в системном стеке при переходе к подпрограмме или прерыванию и восстанавливается при возврате из прерывания. Регистр доступен программно для чтения, записи, а также поразрядной установке или сбросу по маске. Формат регистра PSWR представлен на Рисунок 2.2, а функциональное назначение его полей - в

Таблица 2.1. После системного сброса все разряды регистра слова состояния процессора содержат нули, что отражено на Рисунок 2.2.

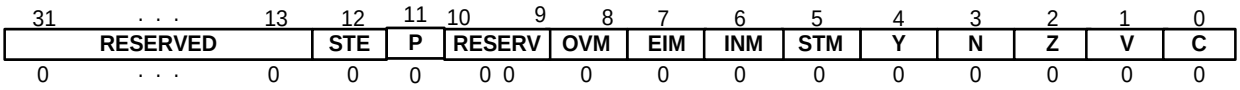


Рисунок 2.2 - Формат регистра PSWR

Таблица 2.1 - Функциональное назначение полей регистра PSWR

Разряд PSWR	Поле	Функция	Описание	
Флаги, формируемые по результатам выполнения скалярных команд				
PSW<0>	C	Признак переноса	0 – 1 –	Флаг сброшен. Флаг установлен.
PSW<1>	V	Признак переполнения	0 – 1 –	Флаг сброшен. Флаг установлен.
PSW<2>	Z	Признак нулевого результата	0 – 1 -	Флаг сброшен. Флаг установлен.
PSW<3>	N	Признак отрицательного результата	0 – 1 –	Флаг сброшен. Флаг установлен.
PSW<4>	Y	Признак заема при выполнении операций умножения по Буту	0 – 1 –	Флаг сброшен. Флаг установлен.
Маски прерываний ядра				
PSW<5>	STM	Маска пошагового прерывания	0 – 1 –	Прерывание маскировано. Прерывание разрешено.
PSW<6>	EXM	Маска внешнего по отношению к ядру прерывания	0 – 1 –	Прерывание маскировано. Прерывание разрешено.
PSW<7>	ERM	Маска прерывания по запрещенной команде	0 – 1 –	Прерывание маскировано. Прерывание разрешено.
PSW<8>	OVM	Маска прерывания по переполнению при выполнении скалярной арифметической операции	0 – 1 –	Прерывание маскировано. Прерывание разрешено.
PSW<10:9>		Зарезервировано		
PSW<11>	P	Разрешение одновременного выполнения нескольких команд	0 – 1 –	Одновременная работа запрещена. Одновременная работа разрешена.
PSW<12>	STE	Разрешение работы аппаратной вершины стека	0 – 1 –	Аппаратная вершина стека отключена. Аппаратная вершина стека включена.
PSW<31:13>		Зарезервировано		

Разряды с 31 по 13 содержат служебную информацию, которая используется для работы аппаратной вершины системного стека, и программно недоступны.

2.4 Регистр запросов на прерывание INTR

32-разрядный регистр запросов на прерывание INTR хранит в себе запросы на прерывания, а также содержит информацию о состоянии некоторых узлов матрично-векторного сопроцессора с фиксированной точкой. Формат регистра INTR представлен на Рисунок 2.3, а функциональное назначение его полей приведено в

Таблица 2.2. На Рисунок 2.3 показано состояние разрядов регистра после системного сброса.

Регистр INTR программно доступен только на чтение и побитовый сброс запросов на прерывание. При возникновении запроса на прерывание в соответствующем бите регистра INTR устанавливается единица, которая сбрасывается, когда процессор входит в подпрограмму обработки данного прерывания либо после программного побитового сброса.

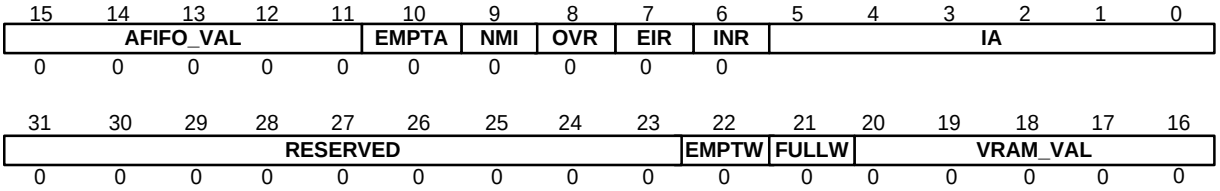


Рисунок 2.3 - Формат регистра INTR

Таблица 2.2 - Функциональное назначение полей регистра INTR

Разряд INTR	Поле	Функция	Описание
INTR<5:0>	IAV[1)]	Номер внешнего прерывания	000000 – Прерывание 0 от сопроцессора арифметики с плавающей точкой (неправильная команда). 000001 – Прерывание 1 от сопроцессора арифметики с плавающей точкой (некорректные данные). 000010 – Прерывание 2 от сопроцессора арифметики с плавающей точкой (переполнение). 000011 – Прерывание 3 от сопроцессора арифметики с плавающей точкой (потеря значимости). 000100 – Прерывание 4 от сопроцессора арифметики с плавающей точкой (потеря точности). 000101 – Прерывание 5 от сопроцессора арифметики с плавающей точкой (потеря данного). 000110 – Прерывание от системного интегратора (обращение блока выборки команд в периферийную область). 000111 Прерывание от моста «системный интегратор – AXI» (ошибка обращения во внешнюю память). 001000 – Прерывание от блока защиты памяти (защита по записи). 001001 – Прерывание от блока защиты памяти (защита по чтению). 001010 – Прерывание от блока таймеров (таймер 0). 001011 – Прерывание от блока таймеров (таймер 1). 001100 – Прерывание 0 от системного контроллера (межпроцессорное прерывание). 001101 – Прерывание 1 от системного контроллера (межпроцессорное прерывание). 001110 – Прерывание 2 от системного контроллера (межпроцессорное прерывание). 001111 – Прерывание 3 от системного контроллера (межпроцессорное прерывание). 010000 – Прерывание 1 от системного контроллера (однократная ошибка при чтении из банка внутренней памяти 0). 010001 – Прерывание 2 от системного контроллера (однократная ошибка при чтении из банка внутренней памяти 1). 010010 – Прерывание 3 от системного контроллера (однократная ошибка при чтении из банка внутренней памяти 2). 010011 – Прерывание от блока сторожевого таймера WDT 010100 – Прерывание 0 от блока интервальных таймеров (DIT). 010101 – Внешнее прерывание 0. 010110 – Внешнее прерывание 1.

Продолжение таблицы 2.2

Разряд INTR	Поле	Функция	Описание	
INTR<5:0>	IAV[1)]	Номер внешнего прерывания	010111 – Прерывание от контроллера USB. 011000 – Прерывание от контроллера SPI. 011001 – Прерывание от контроллера внешней памяти (EMIC). 011010 – Прерывание 1 от блока интервальных таймеров (DIT). 011011 – Прерывание от коммуникационного порта CP0 (передающий канал). 011100 - Прерывание от коммуникационного порта CP0 (принимающий канал). 011101 – Прерывание от коммуникационного порта CP1 (передающий канал). 011110 - Прерывание от коммуникационного порта CP1 (принимающий канал). 011111 - Прерывание от коммуникационного порта CP2 (передающий канал). 100000 - Прерывание от коммуникационного порта CP2 (принимающий канал). 100001 - Внешнее прерывание 2. 100010 - Внешнее прерывание 3. 100011 - Прерывание по завершении работы канала ПДП память-память (от MDMAC). 100100 - Прерывание по ошибке доступа в память канала ПДП память-память (от MDMAC). 100101 - Прерывание по завершении работы канала ПДП память-периферия (от KDMAC). 100110 - Прерывание по завершении работы канала ПДП периферия-память (от KDMAC). 100111 - Прерывание 4 от системного контроллера (межпроцессорное прерывание). 101000 - Прерывание 5 от системного контроллера (межпроцессорное прерывание). 101001 - Прерывание 6 от системного контроллера (межпроцессорное прерывание). 101010 - Прерывание 7 от системного контроллера (межпроцессорное прерывание).	
INTR<6>	EXI	Запрос на внешнее прерывание	0 – 1 –	Отсутствие запроса; Наличие запроса.
INTR<7>	ERI	Запрос на прерывание по запрещенной команде:	0 – 1 –	Отсутствие запроса; Наличие запроса.
INTR<8>	OVI	Запрос на прерывание по переполнению при выполнении скалярной арифметической операции	0 – 1 –	Отсутствие запроса; Наличие запроса.
INTR<9>	MNI	Запрос на немаскируемое внешнее прерывание	0 – 1 –	Отсутствие запроса; Наличие запроса.

Продолжение таблицы 2.2

Разряд INTR	Поле	Функция	Описание	
INTR<10>	AF_E	Признак пустоты AFIFO [2]	0 – 1 –	AFIFO пусто AFIFO не пусто, число записанных слов определяется полем INTR<15:11>
INTR<15:11>	AFIFO_ VAL	Количество слов в AFIFO на момент окончания всех векторных команд, попавших в конвейер процессора на данный момент [2]	0000 – 0001 – . . . 1111 –	1 слово 2 слова . . . 32 слова
INTR<20:16>	VRAM_ VAL	Количество слов в VRAM на момент окончания всех векторных команд, попавших в конвейер процессора на данный момент [2]	0000 – 0001 – . . . 1111 –	1 слово 2 слова . . . 32 слова
INTR<21>	WF_F	Признак полноты WFIFO[2]	0 – 1 –	WFIFO не заполнено полностью WFIFO заполнено полностью
INTR<22>	WF_E	Признак пустоты WFIFO[2]	0 – 1 –	WFIFO не пусто WFIFO пусто
INTR<31:23>	Reserved	Резерв		

[1] Подробно система прерываний описана в разделе 1.5.
[2] AFIFO, VRAM, WFIFO – программно доступные узлы векторно-матричного сопроцессора с фиксированной точкой в процессорной системе NMPU1. Более подробно описаны в главе 4.

3 Матрично-векторный сопроцессор арифметики с плавающей точкой

3.1 Базовые операции матрично-векторного сопроцессора

Базовой операцией сопроцессора арифметики с плавающей точкой является операция умножения 2-элементного вектора на матрицу 2*2, причём элементы вектора и матрицы – 32-разрядные числа с плавающей точкой (m = 0,1):

$$Z_m = \sum_{n=0}^{1} X_n \times W_{n,m} + Y_m$$

Сопроцессор содержит 4 процессорные ячейки, каждое из которых имеет операционное устройство регулярной структуры, состоящее из матрицы умножителей и сумматоров (см. Рисунок 3.1).

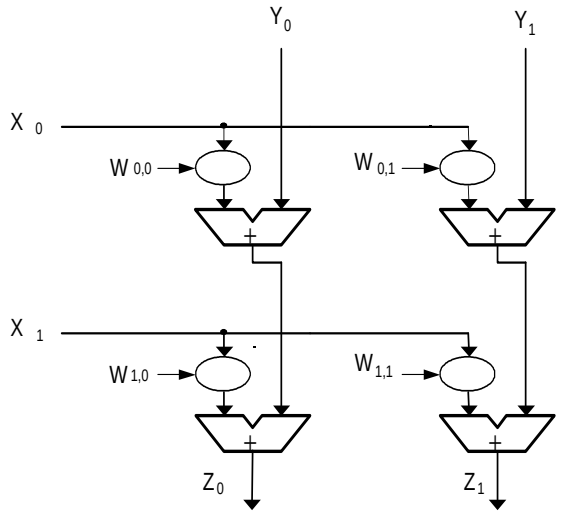


Рисунок 3.1 - Операционное устройство процессорной ячейки

Каждая макроячейка данного устройства выполняет операцию умножения элемента входного вектора Xi на вес Wij, а затем результат прибавляется к выходному значению верхней макроячейки, расположенной в том же столбце. Таким образом, за один такт в каждом столбце независимо вычисляется свой результат.

Поскольку операции производятся над 32-разрядными числами с плавающей точкой, максимальная производительность операционного устройства – 4 умножения и сложения за такт, т.е. 8 FLOPS. Суммарная производительность всех четырёх ячеек сопроцессора – 32 FLOPS. При обработке комплексных чисел за такт также производятся 4 умножения и сложения для одной ячейки и 32 умножения и сложения для всех четырёх процессорных ячеек.

Операционное устройство способно выполнять следующие действия над числами с плавающей точкой, как действительными, так и комплексными, как над данными с одинарной точностью (32 разряда), так и над данными с двойной точностью (64 разряда):

- Умножение матрицы на вектор (матрицу);
- Поэлементное умножение двух векторов;
- Поэлементное сложение/вычитание двух векторов;
- КИХ-фильтр;

- БПФ (только для комплексных чисел).

При работе с числами с плавающей точкой двойной точности достигается максимальная производительность 8 FLOPS.

3.2 Структура матрично-векторного сопроцессора

Матрично-векторный сопроцессор является основным вычислительным узлом процессорного ядра при обработке данных, представленных в формате с плавающей точкой. Структурная схема сопроцессора представлена на Рисунок 3.2.

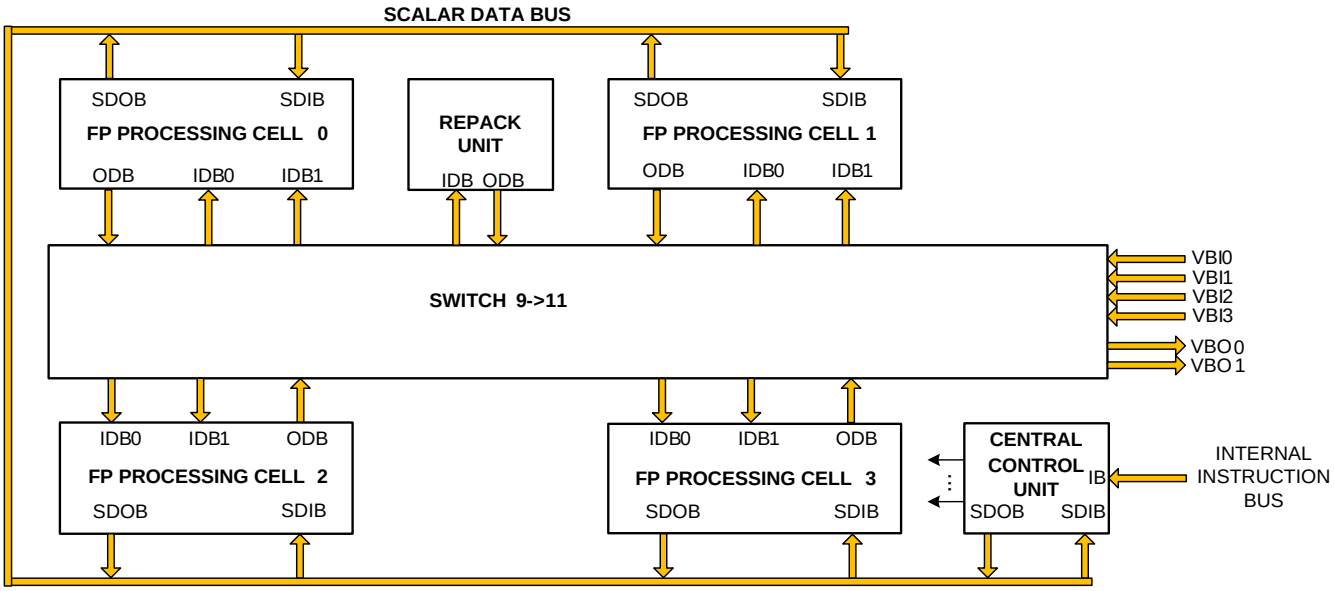


Рисунок 3.2 - Структурная схема матрично-векторного сопроцессора

Основными узлами матрично - векторного сопроцессора являются:

CENTRAL CONTROL UNIT – центральный блок управления сопроцессором. Он принимает команды от управляющего RISC-ядра по внутренней шине команд (INTERNAL INSTRUCTION BUS), дешифрирует и проверяет их на правильность. Если команда правильная, то она запускается на выполнение при условии, что все требуемые ей ресурсы свободны. В случае ошибочности команды она не выполняется, и при этом формируется соответствующее прерывание от сопроцессора. 32-разрядная скалярная шина данных (SCALAR DATA BUS) используется для чтения/записи программно доступных скалярных регистров блока управления.

FP PROCESSING CELL 0, …, FP PROCESSING CELL 3 – 4 одинаковые процессорные ячейки, каждая из которых осуществляет арифметические операции над данными в формате с плавающей точкой как одинарной (32 разряда), так и двойной точности (64 разряда). Каждая ячейка имеет следующие 64-разрядные шины: две входных – IDB0, IDB1 и одну выходную – ODB, что позволяет за один такт осуществить до двух операций чтения и одной операции записи. 32-разрядная скалярная шина данных (SCALAR DATA BUS) используется для чтения/записи программно доступных скалярных регистров процессорных ячеек.

REPACK UNIT – блок упаковки и распаковки данных. Данный блок предназначен для различных преобразований данных в формате с плавающей точкой в целые числа и обратно, а также 64-разрядных данных в 32-разрядные и наоборот.

SWITCH 9->11 – коммутатор 11 в 9. Он позволяет в одном и том же такте осуществить следующие операции:

- До четырёх чтений из памяти по 64-разрядным шинам VBI0 - VBI3 с записью прочитанных данных в процессорные ячейки или в блок упаковки и распаковки данных;
- До двух записей в память по 64-разрядным шинам VBO0 – VBO1 результатов работы процессорных ячеек или блока упаковки и распаковки данных;
- До пяти пересылок данных между процессорными ячейками и блоком упаковки и распаковки данных.

3.3 Форматы векторных данных

3.3.1 Данные

Матрично-векторный сопроцессор предназначен для обработки 32- и 64-разрядных данных в формате с плавающей точкой. При работе с комплексными 32-разрядными данными они объединяются в 64-разрядные слова (32 разряда – действительная часть и 32 разряда – мнимая часть).

3.3.2 64-разрядные слова упакованных данных в памяти

Сопроцессор обменивается с памятью по четырём входным шинам – VBI0 - VBI3– и двум выходным шинам – VBO0 – VBO1. Все обмены осуществляются блоками данных по 64 разряда, причём при работе с действительными 32-разрядными данными в одном 64-разрядном слове упакованы два 32-разрядных числа, при работе с комплексными числами – одно (32 разряда – действительная часть и 32 разряда – мнимая часть). Пример расположения в памяти векторов действительных и комплексных 32-разрядных данных представлен на Рисунок 3.3. Расположение векторов 64-разрядных данных в памяти определяется программистом, и на это нет аппаратных ограничений.

63 32 31 0

D_1	D_0
D_3	D_2
…	...
D_{N+1}	D_N

а) Действительные векторные данные

63 32 31 0

Re_0	Im_0
Re_1	Im_1
…	...
Re_N	Im_N

б) Комплексные векторные данные

Рисунок 3.3 - Представление векторов 32-разрядных данных в памяти

3.3.3 Центральный блок управления сопроцессором

Центральный блок управления матрично-векторным сопроцессором выполняет следующие функции:

- Отслеживает занятость ресурсов сопроцессора;
- Принимает и дешифрирует команды от управляющего RISC-ядра;
- Проверяет команды на правильность. Если проверка прошла успешно, запускает команды на выполнение при условии, что все ресурсы, необходимые ей, свободны. Если нет, вместо команды вставляется код NOP (нет операции), и формируется запрос на прерывание 5 от сопроцессора арифметики с плавающей точкой (неправильная команда);

- Формирует пять запросов на прерывания по результатам работы сопроцессора:
  1) Прерывание 0 от сопроцессора арифметики с плавающей точкой (некорректные данные);
  2) Прерывание 1 от сопроцессора арифметики с плавающей точкой (overflow);
  3) Прерывание 2 от сопроцессора арифметики с плавающей точкой (underflow);
  4) Прерывание 3 от сопроцессора арифметики с плавающей точкой (потеря значимости);
  5) Прерывание 4 от сопроцессора арифметики с плавающей точкой (потеря данных).

Команды, пришедшие в блок управления сопроцессором от управляющего RISC-ядра, считаются неправильными в следующих случаях:

- Задан несуществующий скалярный регистр сопроцессора в командах пересылки типа регистр-регистр или регистр-память;
- Задана арифметическая операция над векторным регистром-источником, в который перед этим отсутствовала запись (т. е. он пуст);
- Задана арифметическая операция над несколькими векторными регистрами, содержащими разное число элементов данных;
- Задана арифметическая операция с маскированием результата, и векторные регистры, определяющие операнды и результат, содержат разное число элементов данных;
- Задана команда записи в память содержимого векторного регистра, в который перед этим отсутствовала запись;
- Задана команда записи в память содержимого векторного регистра, содержащего число элементов данных, отличное от количества повторений, указанных в коде команды;
- Задана любая команда с использованием в качестве источника векторной регистровой пары с разным числом элементов данных в этих регистрах;
- Задана команда пересылки векторных регистров, и регистр-источник пуст;
- Задана команда чтения данных из памяти в блок упаковки и распаковки, и количество пересылаемых данных с учётом типа переупаковки превышает суммарную глубину входного и выходного FIFO этого блока;
- Задана команда записи данных в память из блока упаковки и распаковки, и суммарное число данных во входном и выходном FIFO этого блока меньше количества повторений, указанных в коде команды;
- Задана команда пересылки данных из векторного регистра (пары) в блок упаковки и распаковки, и количество пересылаемых данных с учётом типа переупаковки превышает суммарную глубину входного и выходного FIFO этого блока;
- Задана команда пересылки данных из блока упаковки и распаковки в векторный регистр (пару), и суммарное число данных во входном и выходном FIFO этого блока меньше количества повторений, указанных в коде команды;
- Задана любая команда с использованием в качестве источника векторной регистровой пары с разным числом элементов данных в этих регистрах.

Далее в таблице 3.1 представлен перечень программно доступных скалярных регистров центрального блока управления.

Таблица 3.1 – Перечень программно доступных регистров центрального блока управления

Название	Доступ	Описание
FPCR	ЧТ/ЗП	Регистр управления, содержащий бит параллельной работы и режимы округления
RIER	ЧТ/ЗП	Регистр порядка целого числа при переупаковке
FPIEIR	ЧТ	Регистр, содержащий информацию о прерывании по неправильной команде
FPIOIR	ЧТ	Регистр, содержащий информацию о прерывании по некорректным данным
FPOIR	ЧТ	Регистр, содержащий информацию о прерывании по переполнению
FPUIR	ЧТ	Регистр, содержащий информацию о прерывании по потере значимости
FPIIR	ЧТ	Регистр, содержащий информацию о прерывании по потере точности
FPDLIR	ЧТ	Регистр, содержащий информацию о прерывании по потере данных при сложении трёх операндов

Регистр управления FPCR

Регистр управления FPCR задаёт режим округления результата, а также разрешает или запрещает выполнение нескольких команд в сопроцессоре одновременно. Регистр доступен программно для чтения, поразрядной установки или сброса по маске 0-го разряда, записи заданного значения в 2-1 разряды. Формат регистра FPCR представлен на рисунке 3.4, а функциональное назначение его полей приведено в таблице 3.2. После системного сброса все разряды регистра управления, кроме 0-го, содержат нули, что отражено на рисунке **Ошибка! Источник ссылки не найден.**.

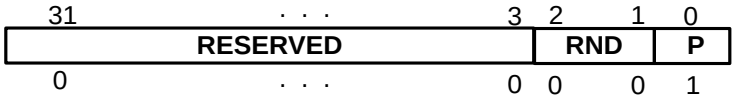


Рисунок 3.4 - Формат регистра FPCR

Таблица 3.2 – Функциональное назначение полей регистра FPCR

Разряд	Обозна-чение	Описание
FPCR <31:3>	RESERVED	Reserved
FPCR <2:1>	RND	Режим округления результата: 00 – к ближайшему; 01 - к -∞; 10 - к +∞; 11 – к нулю.
FPCR <0>	P	Бит параллельной работы: 0 – задаёт строго последовательное выполнение команд сопроцессором; 1 – разрешает одновременное выполнение нескольких команд сопроцессором.

Разряды с 31 по 3 не используются, при их чтении возвращаются все нули, запись в них блокируется.

Регистр порядка целого числа при переупаковке RIER

Регистр порядка целого числа при переупаковке RIER (разряды 10 – 0, причём 10-й разряд - знаковый) задаёт величину предварительного сдвига целого числа, преобразуемого в формат с плавающей точкой. Величина сдвига задаётся в дополнительном коде. Если он равен нулю, при упаковке используются обычные целые числа. Если он – положительное число, производится предварительный сдвиг влево на это число. Если отрицательное – сдвиг вправо на это число. Тем самым поддерживается работа не только с целыми числами, но и с блочной плавающей точкой. Регистр доступен программно для чтения и записи. Формат регистра RIER представлен на рисунке 3.5. После системного сброса все разряды регистра содержат нули, что отражено на рисунке 3.5.

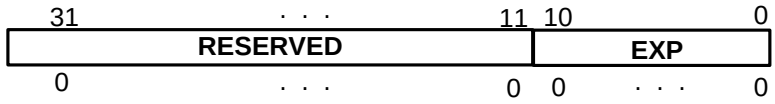


Рисунок 3.5 – Формат регистра RIER

Разряды с 31 по 11 не используются, при их чтении возвращаются все нули, запись в них блокируется.

Регистр, содержащий информацию о прерывании по неправильной команде FPIEIR

Регистр FPIEIR хранит информацию о причине запроса на прерывание от сопроцессора по неправильной команде. Эта информация записывается в регистр в момент обнаружения первой неправильной команды, и будет храниться до тех пор, пока прерывание не будет обработано. Регистр программно доступен только на чтение. Формат регистра FPIEIR представлен на рисунке 3.6, а функциональное назначение его полей приведено в таблице 3.3. После системного сброса все разряды регистра с 31-го по 10-й содержат нули, а с 9-го по 0-й не определены, что и отражено внизу на рисунке 3.6.

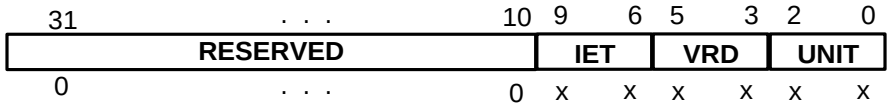


Рисунок 3.6 – Формат регистра FPIEIR

Таблица 3.3 – Функциональное назначение полей регистра FPIEIR

Разряд	Обозна-чение	Описание
FPIEIR <31:10>	RESERVED	Reserved
FPIEIR <9:6>	IET	Тип ошибки в команде, вызвавшей прерывание: 1xxx - векторный регистр-источник пуст; x1xx - несовпадение количества элементов в векторных регистрах-источниках; xx1x – скалярное и векторное чтение из одного и того же векторного регистра-источника; xxx1 – ошибка, относящаяся к блоку упаковки и распаковки данных.
FPIEIR <5:3>	VRD	Номер регистра-приёмника, используемого в команде, не прошедшей проверку.
FPIEIR <2:0>	UNIT	Устройство-приёмник в неправильной команде: 0xx – процессорная ячейка с номером xx; 100 - блок упаковки и распаковки данных; 111 – запись в память.

Разряды с 31 по 10 не используются, при их чтении возвращаются все нули, запись в них блокируется.

Регистры, содержащие информацию о прерываниях по результатам работы сопроцессора FPIOIR, FPOIR, FPUIR, FPIIR и FPDLIR

Регистры FPIOIR, FPOIR, FPUIR, FPIIR и FPDLIR хранят информацию о причине запроса на прерывание от сопроцессора при выполнении им вычислений. Эта информация записывается в регистры в момент обнаружения первого соответствующего события, и будет храниться там до тех пор, пока данное прерывание не будет обработано. Регистры программно доступны только на чтение. Эти регистры имеют один и тот же формат, представленный на рисунке 3.7, а функциональное назначение его полей приведено в таблице 3.4. После системного сброса все разряды регистра с 31-го по 6-й содержат нули, а с 5-го по 0-й не определены, что и отражено внизу на рисунке.

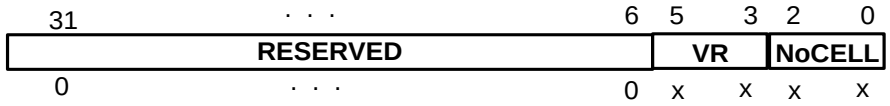


Рисунок 3.7 – Формат регистров FPIOIR, FPOIR, FPUIR, FPIIR и FPDLIR

Таблица 3.4 – Функциональное назначение полей регистров FPIOIR, FPOIR, FPUIR, FPIIR и FPDLIR

Разряд	Обозна-чение	Описание
FPIEIR <31:6>	RESERVED	Reserved
FPIEIR <5:3>	VR	Номер регистра-источника или приёмника, который стал причиной прерывания.
FPIEIR <2:0>	NoCELL	Номер той процессорной ячейки, где произошло прерывание.

Разряды с 31 по 6 не используются, при их чтении возвращаются все нули, запись в них блокируется.

3.3.4 Процессорная ячейка

Структурная схема процессорной ячейки представлена на Рисунок 3.8.

Процессорная ячейка включает в себя следующие основные узлы:

VR0 – VR7 - векторные регистры общего назначения, которые используются как в операциях ввода/вывода, так и в арифметических операциях над данными с плавающей точкой. Их максимальная ёмкость 32 64-разрядных слова упакованных данных. В некоторых типах операций векторные регистры образуют регистровые пары, формируя вектора максимальным размером 32 элемента по 128 разрядов.

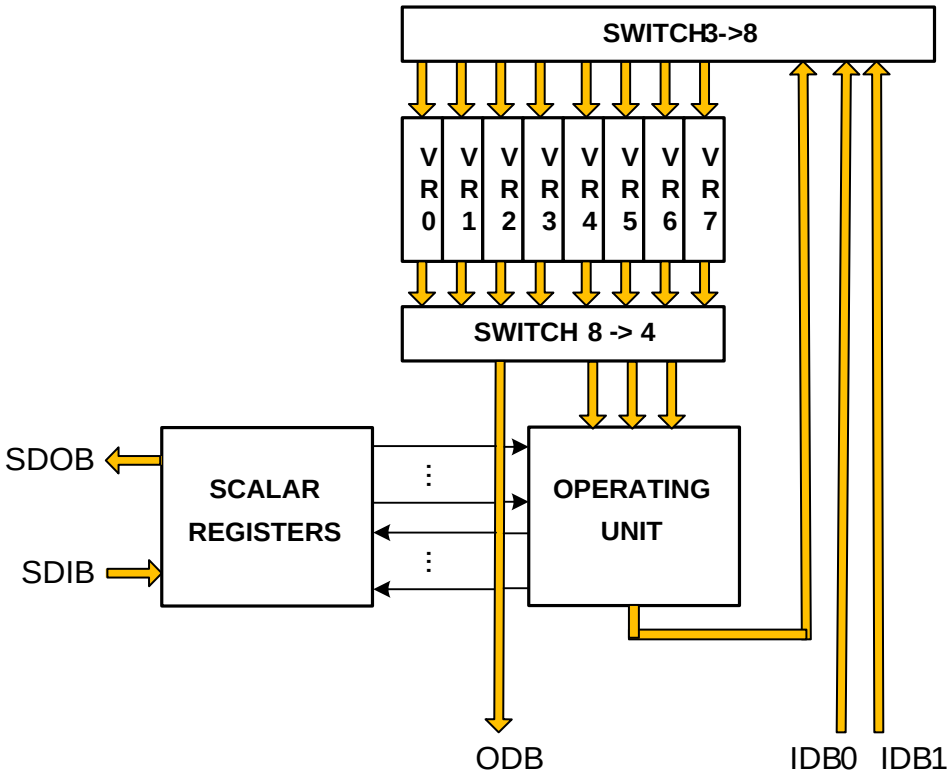


Рисунок 3.8 - Структурная схема процессорной ячейки

SWITCH 3->8 - коммутатор 3 в 8. Он позволяет одновременно записать результат арифметической операции и данные, считанные по шинам IDB0, IDB1 из памяти, а также из других ячеек или блока упаковки и распаковки, в три векторных регистра соответственно.

SWITCH 8->4 - коммутатор 8 в 4. Он предназначен для выбора до трех источников для арифметических операций, а также одного источника для записи в память, в другие ячейки или в блок упаковки и распаковки по выходной шине ODB. Причём, в качестве этих источников может использоваться любой из восьми векторных регистров.

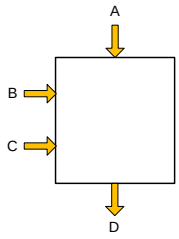
OPERATING UNIT – операционное устройство для выполнения векторных и матричных операций над данными в формате с плавающей точкой. Данное устройство может работать в одном из четырёх режимов (см. Рисунок 3.9):

- Операции над данными в формате с плавающей точкой двойной точности - см. Рисунок 3.9 а). В этом режиме все входные операнды A, B и C и результат D представляют собой 64-разрядные числа в формате с плавающей точкой двойной точности. Основная операция режима – умножение двух чисел и сложение с третьим;
- Операции над комплексными данными в формате с плавающей точкой одинарной точности - см. Рисунок 3.9 б). Данный режим характеризуется тем, что все входные операнды: A1 и A0, B1 и B0, C1 и C0, а также результат D1 и D0 представляют собой комплексные 64-разрядные числа, причём старшие 32 разряда содержат действительную часть (A1, B1, C1 и D1), а младшие 32 разряда (A0, B0, C0 и D0) – мнимую часть. Основная операция режима – умножение двух комплексных чисел и сложение с третьим;
- Операции над векторными данными в формате с плавающей точкой одинарной точности - см. Рисунок 3.9 в). В этом режиме все входные операнды A1 и A0, B1 и B0, C1 и C0, а также результат D1 и D0 образуют 64-разрядные вектора из двух

32-разрядных элементов. Основная операция режима – умножение двух векторов по два элемента и сложение с третьим двухэлементным вектором;

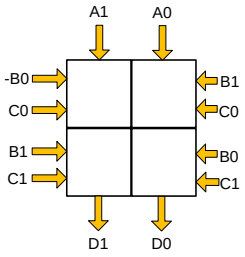
- Операции над матричными данными в формате с плавающей точкой одинарной точности - см. Рисунок 3.9 г). Данный режим характеризуется тем, что все входные операнды A1 и A0, B1 и B0, B3 и B2, C1 и C0, а также результат D1 и D0 образуют 64-разрядные вектора из двух 32-разрядных элементов, как и в предыдущем случае. Основная операция режима – умножение вектора из двух элементов [C1 C0] на матрицу 2*2 элемента $\begin{bmatrix} B1 & B3 \\ B0 & B2 \end{bmatrix}$ и сложение с третьим двухэлементным вектором [A1 A0]. Особенностью данной операции является то, что матрица считывается сразу из пары векторных регистров: из того, что указан в команде (он обязан иметь чётный номер) считывается столбец $\begin{bmatrix} B1 \\ B0 \end{bmatrix}$, а из регистра с номером на единицу больше – столбец $\begin{bmatrix} B3 \\ B2 \end{bmatrix}$. Имеется также возможность в зависимости от кода команды сформировать матрицу по строкам: из чётного регистра считается строка [B0 B2], из нечётного – строка [B1 B3]. В первом случае это позволяет осуществить умножение матрицы B на вектор-строку C слева, во втором - умножение матрицы B на вектор-столбец C справа.

  Остальные операнды в виде одного 64-разрядного данного или вектора из двух 32-разрядных данных считываются только из одного векторного регистра. Результат также пишется только в один векторный регистр.



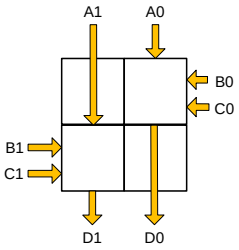
D = A + B*C

а) Операции над данными в формате с плавающей точкой двойной точности



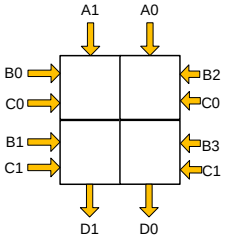
D0 = A0 + B0*C1 + B1*C0
D1 = A1 + B1*C1 – B0*C0

б) Операции над комплексными данными в формате с плавающей точкой одинарной точности



D0 = A0 + B0*C0
D1 = A1 + B1*C1

в) Операции над векторными данными в формате с плавающей точкой одинарной точности



D0 = A0 + B2*C0 + B3*C1
D1 = A1 + B0*C0 + B1*C1

г) Операции над матричными данными в формате с плавающей точкой одинарной точности

Рисунок 3.9 - Режимы работы операционного устройства

SCALAR REGISTERS – программно доступные скалярные регистры, входящие в состав каждой процессорной ячейки. В Таблица 3.5 представлен их перечень.

Таблица 3.5 – Перечень программно доступных регистров процессорной ячейки

Название	Доступ	Описание
FPFR	ЧТ/ЗП	Регистр флагов
SPMRL	ЧТ/ЗП	Младшая часть регистра маски одинарной точности
SPMRH	ЧТ/ЗП	Старшая часть регистра маски одинарной точности
DPMR	ЧТ/ЗП	Регистр маски двойной точности

Регистр флагов FPFR

Регистр FPFR хранит флаги результата вычисления последнего элемента результата последней выполненной арифметической команды (для 32-разрядных данных соответствует результату в разрядах 63-32). Регистр программно доступен на чтение и запись. Формат регистра FPFR представлен на рисунке 3.10, а функциональное назначение его полей приведено в таблице 3.6. После системного сброса все разряды регистра содержат нули, что и отражено внизу на рисунке.

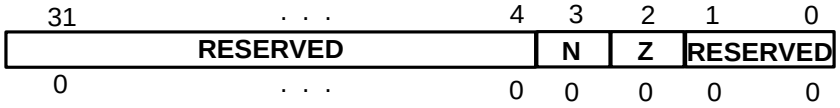


Рисунок 3.10 – Формат регистра FPFR

Таблица 3.6 – Функциональное назначение полей регистра FPFR

Разряд	Обозна-чение	Описание
FPFR <31:4>	RESERVED	Reserved
FPFR <3>	N	Признак отрицательного знака результата
FPFR <2>	Z	Признак нулевого результата
FPFR <1:0>	RESERVED	Reserved

Разряды с 31-го по 4-й и с 1-го по 0-й не используются, при их чтении возвращаются все нули, запись в них блокируется.

Регистры масок одинарной точности SPMRL (младшая часть) и SPMRH (старшая часть)

В качестве альтернативы работы с условными командами введён механизм работы с масками. В общем случае, маска формируется по результату арифметической команды. Каждому i-му элементу вектора результата (i = 0,1,…,N, где N≤31) ставится в соответствие i-й разряд регистра маски. Если для i-го элемента вектора результата выполняется заданное программистом единое для всего вектора условие, то в i-й разряд регистра маски пишется единица, иначе – ноль. В дальнейшем, при выполнении арифметических операций есть возможность замаскировать результат по содержимому регистра маски: если разряд регистра маски равен 1, то соответствующий элемент вектора результата запишется в регистр-приёмник, если нет - соответствующий элемент в регистре-приёмнике меняться не будет. Таким образом, использование механизма масок позволяет обойтись без команд условного перехода.

Поскольку арифметические операции в формате с плавающей точкой одинарной точности (32 разряда) требуют максимум 64 бита маски (32 элемента вектора по два 32-разрядных слова), используются два 32-разрядных регистра маски одинарной точности – SPMRL (младшая часть) и SPMRH (старшая часть). Их форматы приведены, соответственно, на рисунках 3.11 и 3.12. 0-й разряд регистра SPMRL содержит маску, соответствующую 0-му

элементу вектора результата, разрядам с 31-го по 0-й. Соответственно, 0-й разряд регистра SPMRH содержит маску, соответствующую 0-му элементу вектора результата, разрядам с 63-го по 32-й. Таким образом, SPMRL содержит маски для элементов вектора с 0-го по 31-й только для младших 32-х разрядов результата, а SPMRH - маски для элементов вектора с 0-го по 31-й только для старших 32-х разрядов результата. Если число элементов в векторе результата меньше 32-х, то старшие разряды регистров масок не используются.

Регистры SPMRL и SPMRH программно доступны на чтение и запись. После системного сброса все разряды регистров не определены, что и отражено на Рисунок 3.11 и Рисунок 3.12.

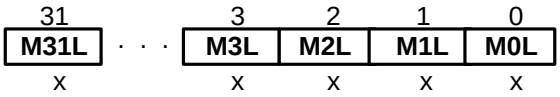


Рисунок 3.11 – Формат регистра SPMRL

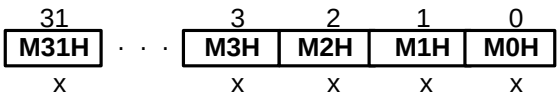


Рисунок 3.12 – Формат регистра SPMRH

При выполнении арифметических операций в формате с плавающей точкой двойной точности (64 разряда) требуется максимум 32 бита маски (32 элемента вектора по одному 64-разрядному слову). В этом случае используется только один регистр маски, а именно SPMRH.

Регистр маски двойной точности DPMR

Регистр DPMR является псевдорегистром. Запись в него вызовет запись одних и тех же 32-разрядных данных в регистры SPMRL и SPMRH, чтение из него вызовет чтение SPMRH. Данный псевдорегистр используется вместе с арифметическими операциями в формате с плавающей точкой двойной точности, чтобы программист не заботился о том, какой из регистров - SPMRL или SPMRH – используется. Вместе с тем, для такого вида операций можно использовать регистр SPMRH в явном виде.

3.3.5 Блок упаковки и распаковки данных

Структурная схема блока упаковки и распаковки данных представлена на Рисунок 3.13.

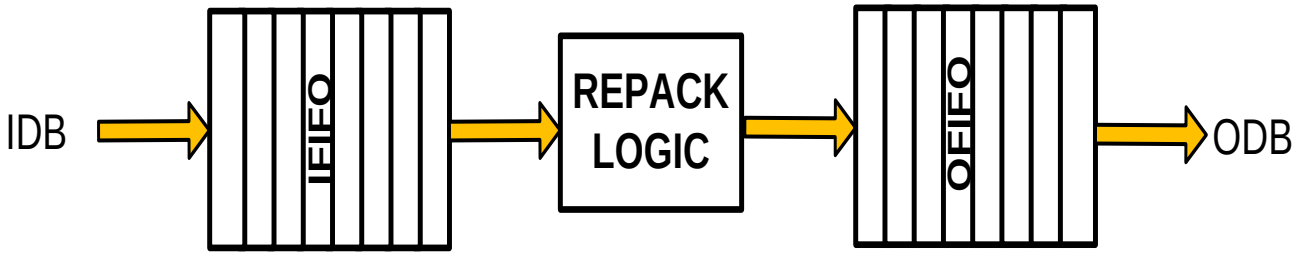


Рисунок 3.13 - Структурная схема блока упаковки и распаковки данных

Блок упаковки и распаковки данных включает в себя следующие основные узлы:

IFIFO – входное FIFO, куда данные, считанные по шине IDB из памяти или из одной из процессорных ячеек, записываются для дальнейшего преобразования.

OFIFO – выходное FIFO, откуда результат переупаковки считывается для записи в память, в другие ячейки или в блок упаковки и распаковки по выходной шине ODB.

REPACK LOGIC – комбинационная схема, реализующая требуемую упаковку и распаковку данных. На её входе и выходе возможно использование следующих восьми форматов данных, причём возможно любое их сочетание:

- В блоке 64-разрядных данных упакованы два 32-разрядных числа в формате с фиксированной точкой;
- В блоке 64-разрядных данных упакованы два 32-разрядных числа в формате с фиксированной точкой, но в дальнейшем используются только старшие 32 разряда (младшие 32 разряда обнуляются);
- В блоке 64-разрядных данных упакованы два 32-разрядных числа в формате с фиксированной точкой, но в дальнейшем используются только младшие 32 разряда (старшие 32 разряда обнуляются);
- В блоке 64-разрядных данных находится одно число в формате с фиксированной точкой;
- В блоке 64-разрядных данных упакованы два 32-разрядных числа в формате с плавающей точкой;
- В блоке 64-разрядных данных упакованы два 32-разрядных числа в формате с плавающей точкой, но в дальнейшем используются только старшие 32 разряда (младшие 32 разряда обнуляются);
- В блоке 64-разрядных данных упакованы два 32-разрядных числа в формате с плавающей точкой, но в дальнейшем используются только младшие 32 разряда (старшие 32 разряда обнуляются);
- В блоке 64-разрядных данных находится одно число в формате с плавающей точкой двойной точности.

Таким образом, возможны 64 варианта упаковки и распаковки данных, имеющих различные форматы и разрядности.

4 Матрично-векторный сопроцессор для обработки данных, представленных в формате с фиксированной точкой и программируемой разрядностью

4.1 Базовые операции матрично-векторного сопроцессора

Архитектура матрично-векторного сопроцессора даёт уникальную возможность варьировать между производительностью и точностью вычислений для базовой процедуры:

$$Y_m = U_m + \sum_{n=1}^{N} X_n \times W_{n,m}$$

В зависимости от необходимости можно выбрать необходимую разрядность входных данных и результата (точность вычислений). Число умножений и сложений (MAC), выполняемых за один такт, зависит от разрядности операндов. Наибольшая производительность – 224 MAC – достигается при работе с 2-разрядными операндами. Имеется возможность поднять точность вычислений, если увеличить разрядность операндов до 32-х. В этом случае достигается производительность 2 MAC с получением 64-разрядного результата. Векторный сопроцессор содержит операционное устройство регулярной структуры, похожее на матричный умножитель (см. Рисунок 4.1).

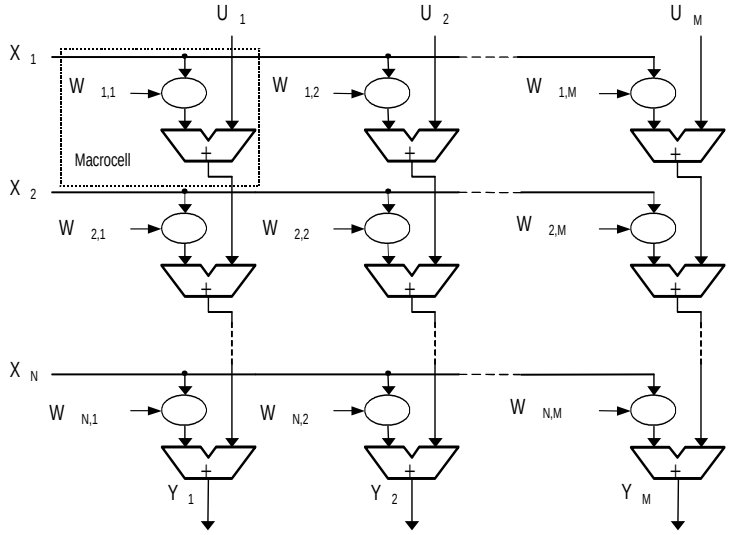


Рисунок 4.1 - Операционное устройство матрично-векторного сопроцессора

Данное устройство состоит из ячеек, содержащих 1-разрядную память (триггер) и некоторую комбинационную логику. Пользователь может поделить матрицу ячеек на макроячейки, используя два программно доступных 64-разрядных конфигурационных регистра. Эти регистры задают границы между строками и столбцами макроячеек таким образом, что каждая макроячейка выполняет операцию умножения элемента входного вектора Xi на заранее загруженный вес Wij, а затем результат прибавляется к выходному значению верхней макроячейки, расположенной в том же столбце. Таким образом, за один такт в каждом столбце независимо вычисляется свой результат. Пример конфигурации векторного узла для работы с 8-разрядными входными данными (Xi) и весами (Wij) приведён на Рисунок 4.2. В этом случае достигается пиковая производительность в 24 MAC (за один процессорный такт производятся 24 операции умножений с накоплением с получением 21-разрядных результатов).

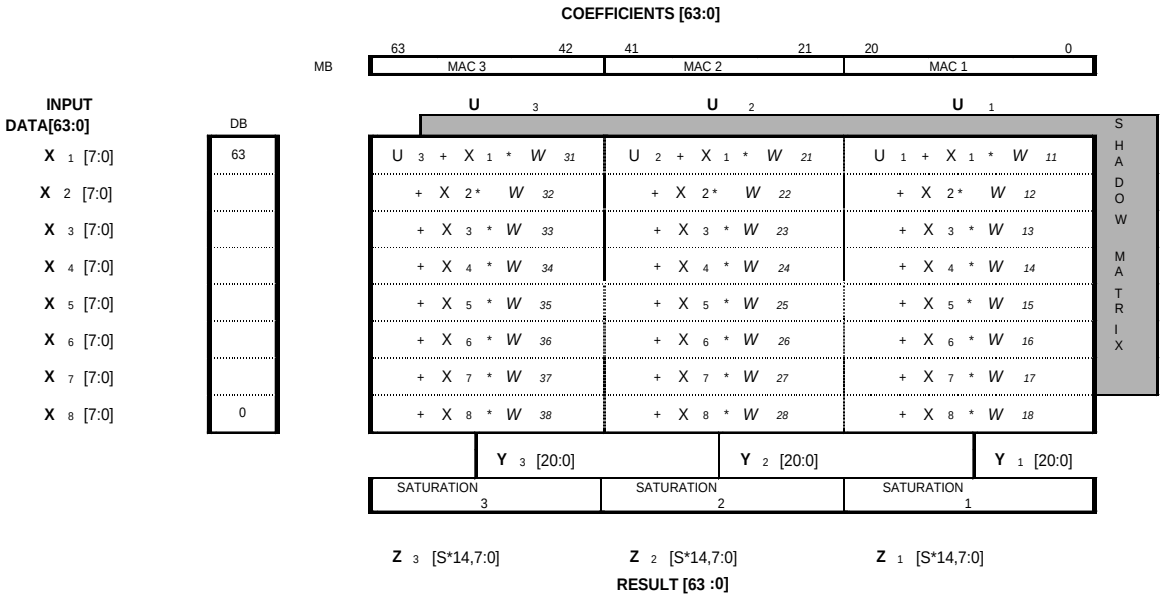


Рисунок 4.2 - Пример работы векторного сопроцессора с 8-разрядными входными данными и весами

Число операций умножений с накоплением зависит от разрядности входных операндов и весов. Конфигурация сопроцессора может меняться динамически в процессе вычислений. Можно начать вычисления с небольшой разрядностью и с большой производительностью, а затем по мере накопления разрядности в промежуточных результатах перейти к обработке данных большей разрядности за счёт снижения быстродействия. На Рисунок 4.3 можно наглядно видеть, как зависит производительность от разрядности входных операндов и весов.

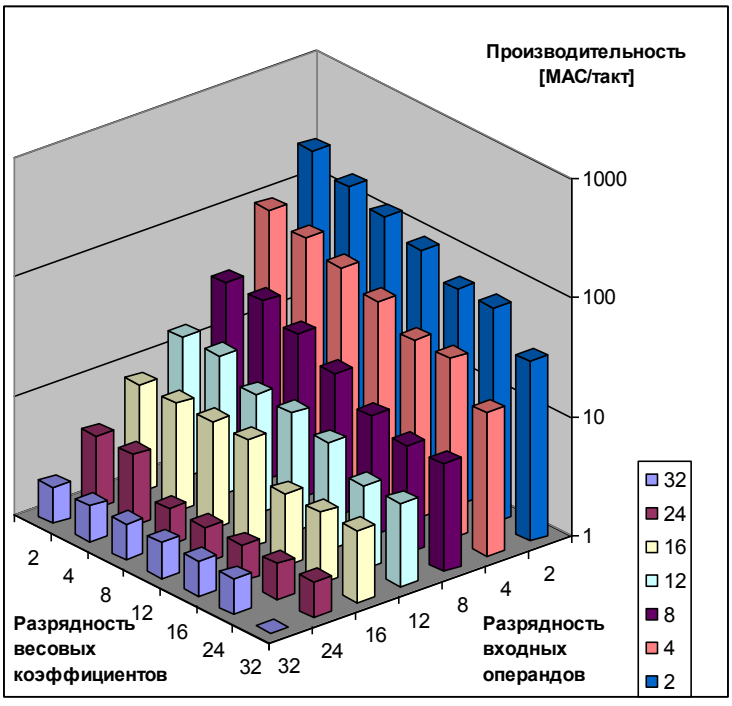


Рисунок 4.3 - Зависимость производительности от разрядности входных данных

Операционное устройство достигает ещё большей производительности при выполнении булевого умножения, когда разрядность входных операндов и весов равна 1. В этом случае она равна 1024 MOPS за один процессорный такт. Имеется ещё одно интересное свойство использования 1-разрядных коэффициентов. В этом случае операционное устройство превращается в мощный коммутатор, когда перестановка битов в 64-разрядном входном операнде происходит за один такт.

Загрузка новых весовых коэффициентов в операционное устройство осуществляется за столько тактов, сколько требуется загрузить строк весовых коэффициентов, т. е. от 1 до 32 тактов. Чтобы скомпенсировать потери времени при изменении весов, используется теневая матрица. Новые коэффициенты грузятся в теневую матрицу в фоновом режиме и затем за один такт переписываются в рабочую.

Для предотвращения переполнения аппаратно реализована функция насыщения над 64-разрядными словами упакованных данных (см. Рисунок 4.4), причём границы насыщения задаются с помощью программно доступных регистров управления этой функцией. Функция насыщения не уменьшает разрядность входных операндов, но уменьшает число значащих разрядов в каждом элементе входного вектора.

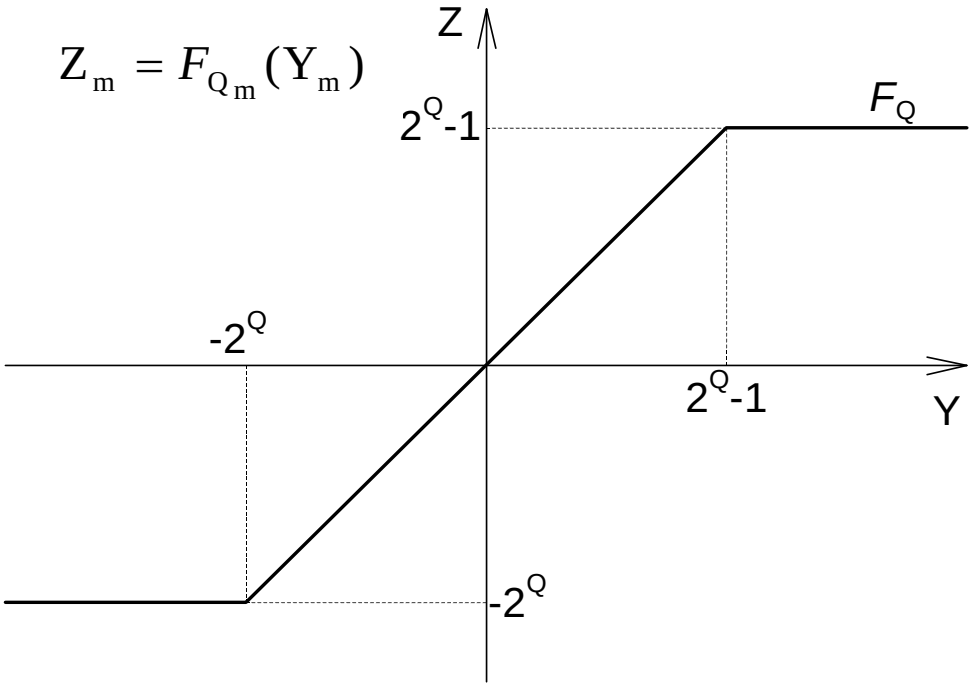


Рисунок 4.4 - Функция насыщения

4.2 Структура матрично-векторного сопроцессора

Матрично-векторный сопроцессор является основным вычислительным узлом процессорного ядра. Сопроцессор ориентирован на обработку данных произвольной разрядности от 1 до 64 разрядов, упакованных в 64-разрядные слова. Структурная схема сопроцессора представлена на

Рисунок 4.5.

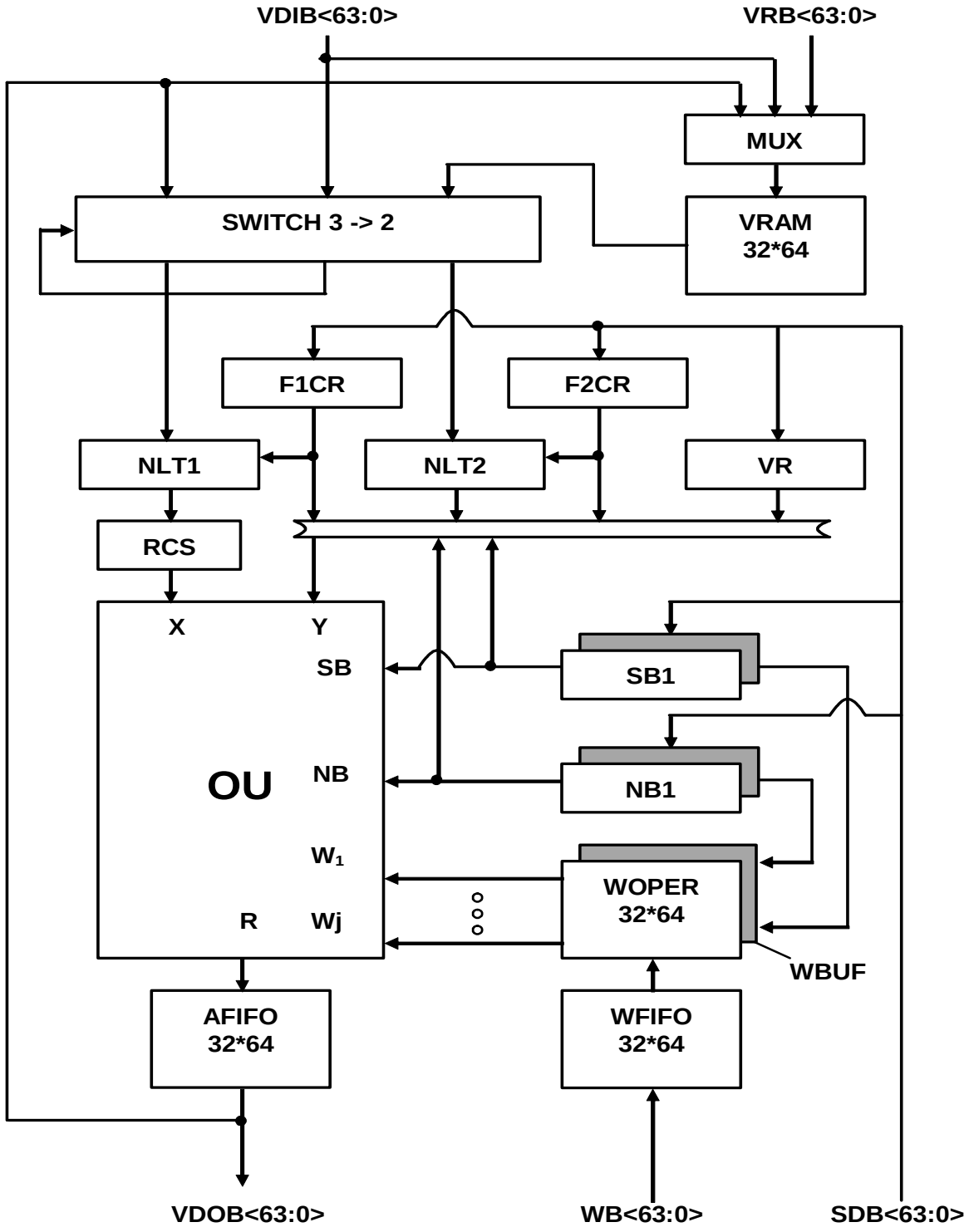


Рисунок 4.5 - Структурная схема матрично-векторного сопроцессора

Основными узлами матрично- векторного сопроцессора являются:

Операционное устройство	OU
Циклический сдвигатель вправо	RCS
Нелинейные преобразователи	NLT1, NLT2
Коммутатор 3 в 2	SWITCH 3→2
Память весовых коэффициентов и их операционная память	WBUF и WOPER
FIFO весовых коэффициентов	WFIFO
Накопительное FIFO	AFIFO
Векторный регистр	VRAM
Регистр порогов	VR
Регистры управления нелинейными преобразователями	F1CR, F2CR
Регистр границ и операционный регистр границ операнда X	SB1 и SB2
Регистр границ операнда Y и результата R	NB1 и NB2

Работа всех перечисленных выше узлов будет описана в следующих подразделах.

4.3 Форматы векторных данных

4.3.1 Данные

Матрично-векторный сопроцессор предназначен для обработки целочисленных данных, разрядность которых может быть произвольной и в общем случае лежит в диапазоне от 1 до 64 разрядов. Однако, схемотехническая реализация ряда исполнительных узлов сопроцессора накладывает дополнительные ограничения на разрядность данных:

- Множители, поступающие на вход X OU и используемые в операциях взвешенного суммирования, должны иметь четную разрядность;
- Переменные, для которых вычисляется нелинейная функция активации, должны иметь не менее двух разрядов.

Целочисленные данные, используемые в арифметических операциях, должны быть представлены в дополнительном коде. Результаты арифметических операций формируются также в дополнительном коде.

4.3.2 64-разрядные слова упакованных данных

В каждом такте сопроцессор осуществляет обработку всех данных слова, поступающего на любой из его исполнительных узлов. При этом исполнительные узлы определяют границы данных в слове по содержимому соответствующих конфигурационных регистров. Разряды регистра NB2, в которых записана 1, соответствуют старшим разрядам данных слова, поступающего на вход Y, а при выполнении арифметических операций и на вход X OU. Разряды регистра SB2, в которых записана 1, соответствуют младшим разрядам данных слова, поступающего на вход X OU при выполнении операции взвешенного суммирования. Если в i-м разряде регистра F1CR записана 1, а в (i+1)-м - 0, то i-й разряд слова, поступающего на вход преобразователя NLT1, будет являться старшим разрядом одного из данных, упакованных в этом слове. Аналогично регистр F2CR определяет границы данных в слове, поступающем на вход преобразователя NLT2. В качестве примера на Рисунок 4.6 приведено содержимое конфигурационных регистров для слова, состоящего из двухразрядного операнда D_1, четырехразрядного операнда D_2, шестиразрядного операнда D_3 и т. д. Здесь ‘x’ в NLTCR1 и NLTCR2 означает, что значение данного разряда не определяет границ данных в слове.

13	12	11	10	9	8	7	6	5	4	3	2	1	0	
D3								D2				D1		- слово упакованных данных
1	0	0	0	0	0	0	0	1	0	0	0	1	0	- NB2
0	0	0	0	0	0	0	1	0	0	0	1	0	1	- SB2
1	x	x	x	x	x	x	0	1	x	x	0	1	x	- F1CR (F2CR)

Рисунок 4.6 - Форматы конфигурационных регистров

В регистры NB2 и SB2 информация поступает соответственно из регистров NB1 и SB1 при выполнении команды LOAD. Регистры NB1, SB1, F1CR F2CR и VR подключены к скалярной шине данных процессорного ядра – SDB<63:0> - и программно доступны по записи.

4.3.3 Матрицы весовых коэффициентов и вектора слов упакованных данных

Все исполнительные узлы сопроцессора позволяют за один такт выполнять операции над 64-разрядными словами упакованных данных. Базовой операцией является операция взвешенного суммирования, при выполнении которой в каждом такте используется матрица весовых коэффициентов, хранящаяся в WOPER. Размерность матрицы весовых коэффициентов может достигать 32 64-разрядных слов упакованных весовых коэффициентов.

С целью эффективного использования аппаратных ресурсов сопроцессора предусмотрен механизм подкачки новой матрицы весовых коэффициентов из внешней памяти на фоне выполнения операций с текущим содержимым WOPER. 64-разрядные слова упакованных весовых коэффициентов поступают с одной из внешних шин процессорного ядра – WB<63:0> - через WFIFO в WBUF. При этом процесс загрузки теневой матрицы WBUF длится от 1 до 32 тактов в зависимости от того, сколько грузится слов упакованных весовых коэффициентов, и управляется содержимым регистров SB1 и NB1, которые задают соответственно количество слов упакованных весовых коэффициентов в матрице весов и границы весовых коэффициентов в этих 64-разрядных словах. Затем по команде LOAD за один такт содержимое теневой матрицы WBUF переписывается в рабочую матрицу ввесов WOPER, а содержимое регистров NB1 и SB1 - в регистры NB2 и SB2 соответственно. Одновременно с этим OU может выполнять операции с использованием прежнего содержимого WOPER, NB2 и SB2.

Так как процесс загрузки матрицы весов длится от 1 до 32 тактов, то каждая векторная команда позволяет выполнять за L тактов одни и те же операции над векторами из L 64-разрядных слов упакованных данных. Переменная L задается в одном из полей команды и может принимать целочисленное значение в диапазоне от 1 до 32.

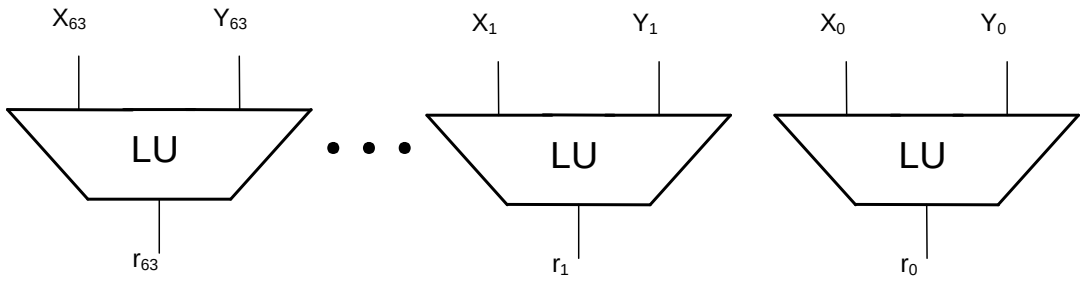
4.4 Операционное устройство OU

OU служит для выполнения арифметических и логических операций над 64-разрядными словами упакованных данных **X**={X_K ... X_1} и **Y**={Y_I ... Y_1}, поступающими соответственно на входы X и Y OU, и матрицей весов **W**, которая подается на входы W_1,...,W_J в виде J ой операции: K=I - для арифметических операций, K=J - для операции взвешенного суммирования.

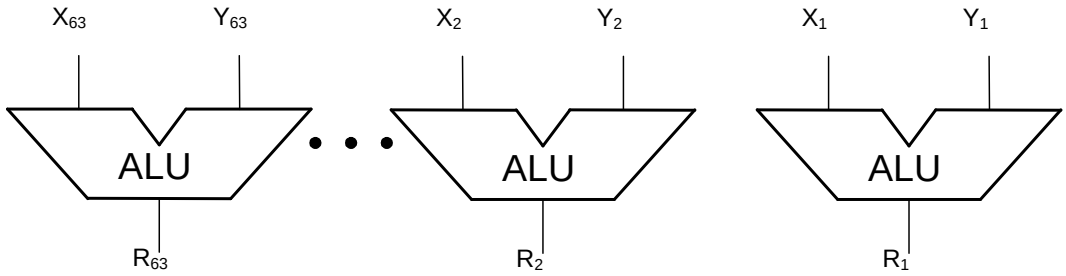
Тип выполняемой операции задается кодом команды. Операции выполняются в конвейере с темпом одна операция за такт. Никаких признаков результатов операций OU не формирует.

OU выполняет три следующих набора операций, каждому из которых соответствует своя программная модель OU, как показано на Рисунок 4.7:

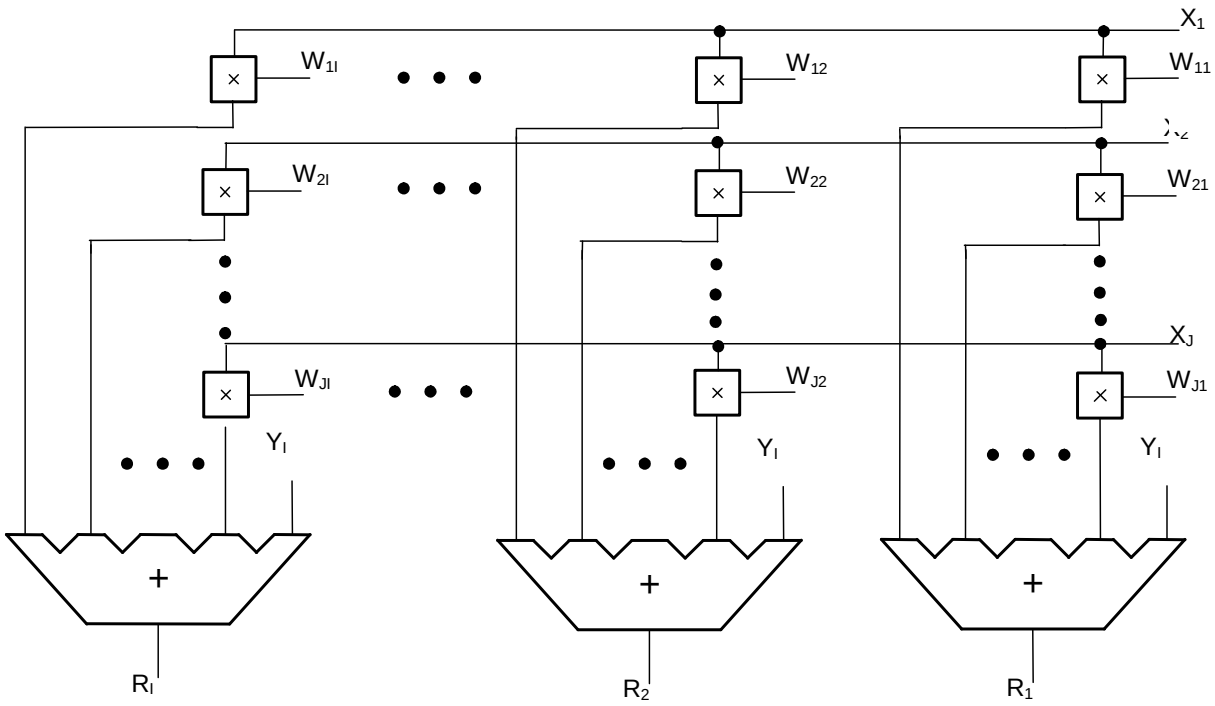
- 16 всевозможных поразрядных логических операций над 64-разрядными операндами, поступающими на входы X и Y. В данном режиме OU представляется в виде 64 параллельно работающих одноразрядных логических устройств LU, выполняющих одну и ту же логическую операцию над каждой парой разрядов входных операндов (Рисунок 4.7 а). Содержимое регистров SB2 и NB2 не влияет на результаты логических операций.
- Арифметические операции над словами X и Y: Xi + Yi; Xi - Yi ; Xi + 1; Xi - 1, где i=1,...,I. При выполнении данных арифметических операций OU можно представить в виде I параллельно включенных арифметических устройств AU, выполняющих одну и ту же арифметическую операцию над соответствующими данными слов X и Y (Рисунок 4.7 б). Количество и разрядности AU и соответствующих данных в словах X, Y и R совпадают и определяются содержимым регистра NB2. Содержимое регистра SB2 не влияет на результаты данных арифметических операций.
- Сложение вектора Y с произведением матрицы весов W : Ri = Yi + Σ Wij * Xj , где i=1,...,I, j=1,...,J. При выполнении данной операции, получившей также название взвешенного суммирования, OU можно представить в виде I параллельно работающих схем, каждая из которых содержит J умножителей и один (J+1)- операндный сумматор (Рисунок 4.7 в). Количество I и разрядности соответствующих операндов в словах Y, WJ,...,W1 и R совпадают и определяются содержимым регистра NB2, а количество J и разрядности операндов в векторе X определяются содержимым регистра SB2. Умножители в OU реализованы по модифицированному алгоритму Бута, поэтому разрядность каждого операнда, входящего в состав слова X, должна быть четной.



а) Программная модель OU при выполнении логических операций.



б) Программная модель OU при выполнении арифметических операций.



в) Программная модель OU при выполнении взвешенного суммирования.

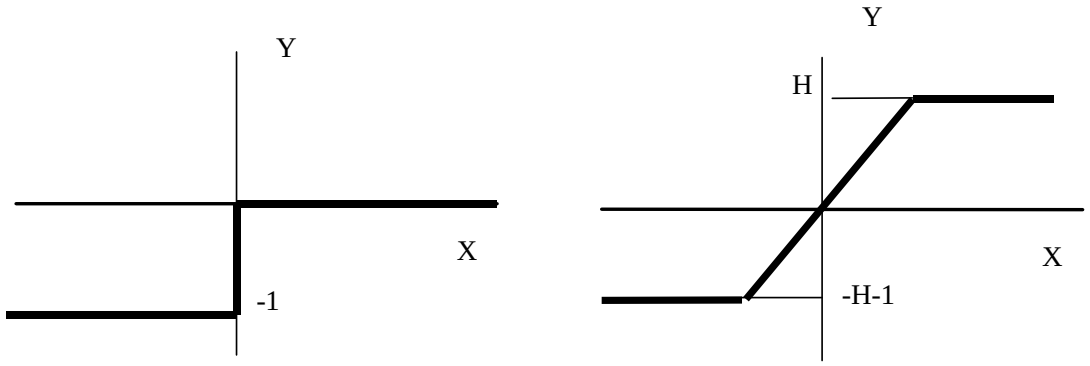
Рисунок 4.7 - Программные модели OU в различных режимах его работы

4.5 Циклический сдвигатель вправо RCS

В зависимости от кода команды 64-разрядные слова, поступающие на вход X OU, проходят через RCS без изменений или циклически сдвигаются вправо на один разряд. За один такт выполняется сдвиг одного слова как единого операнда, независимо от количества данных в слове.

4.6 Нелинейные преобразователи NLT1, NLT2

NLT1 и NLT2 служат для вычисления нелинейных функций активации над 64-разрядными словами упакованных данных. Для каждого нелинейного преобразователя в VU предусмотрен свой программно доступный регистр управления (F1CR для NLT1 и F2CR для NLT2), содержимое которого в зависимости от кода выполняемой команды может определять количество и разрядности данных, составляющих обрабатываемое слово, а также максимальное абсолютное значение результата вычисления функции насыщения H для каждого составляющего слова (Рисунок 4.8).



а) Пороговая функция б) Функция насыщения

Рисунок 4.8 - Нелинейные функции активации, вычисляемые NLTx

Преобразователи NLT1 и NLT2 абсолютно идентичны, поэтому для обозначения нелинейного преобразователя далее будет использоваться мнемоника NLTx. Все сказанное о NLTx будет относиться как к NLT1, так и к NLT2. Аналогично, мнемоника FxCR используется ниже для обозначения одного из регистров F1CR или F2CR.

В зависимости от кода команды NLTx выполняет за один такт одну из трех операций над 64-разрядным словом упакованных данных:

- Пропускает на выход входное слово без изменений независимо от содержимого регистра FxCR;
- Вычисляет пороговую функцию для каждого операнда входного слова (Рисунок 4.8 а). При этом содержимое регистра FxCR определяет количество и разрядности данных входного и выходного слова. Формат регистра FxCR при вычислении пороговой функции представлен на Рисунок 4.6;
- Вычисляет функцию насыщения для каждого операнда входного слова (Рисунок 4.8 б). При этом содержимое регистра FxCR определяет не только количество и разрядности операндов входного и выходного слов упакованных данных, но и максимальные абсолютные значения каждого операнда выходного слова. Формат регистра FxCR при вычислении функции насыщения представлен на Рисунок 4.6.

Из представленных на Рисунок 4.6 форматов регистра FxCR следует, что минимальная разрядность операндов, составляющих обрабатываемое слово, равна двум. Поэтому за один такт NLTx может обрабатывать от 1 до 32 данных различной разрядности, суммарная разрядность которых равна 64.

Необходимо отметить, что при прохождении слова упакованных данных через NLTx независимо от выполняемой им операции количество и разрядности операндов, составляющих 64-разрядное слово, не изменяются. Уменьшение абсолютных значений входных операндов в результате вычисления нелинейных функций активации сопровождается расширением знаковых разрядов операндов. Таким образом, NLTx служит только для уменьшения абсолютных значений входных операндов, а не для уменьшения их разрядности.

4.7 Памяти весовых коэффициентов WBUF и WOPER

Память весовых коэффициентов состоит из двух матриц ячеек памяти WBUF и WOPER, каждая из которых имеет емкость 32x64 бита и позволяет хранить матрицу весов **W** в виде J 64-разрядных слов упакованных весовых коэффициентов: $\mathbf{W_1}=\{W_{1I} \ldots W_{11}\},\ldots,$ $\mathbf{W_J}=\{W_{JI} \ldots W_{J1}\}$.

WOPER служит для хранения матрицы весов, используемой в операциях взвешенного суммирования, выполняемых OU. Выходы всех ячеек WOPER соединены непосредственно с соответствующими входами OU, а их входы - с выходами соответствующих ячеек WBUF. Благодаря этому запись во все ячейки WOPER осуществляется за один такт по команде LOAD. При этом содержимое WBUF целиком копируется в WOPER. Одновременно с этим содержимое регистров SB1 и NB1 копируется в регистры SB2 и NB2.

WBUF служит для подкачки из WFIFO новой матрицы весов на фоне выполнения OU текущих операций взвешенного суммирования с использованием старой матрицы весов, хранящейся в WOPER. Загрузка матрицы весов в WBUF инициируется одной командой и осуществляется за J тактов путем последовательной записи J 64-разрядных слов упакованных весовых коэффициентов, выбираемых из WFIFO. При этом количество загружаемых слов J определяется содержимым SB1 (J равно количеству единиц в SB1). В режиме записи WBUF работает по аналогии с памятью магазинного (стекового) типа. Причем первым загружается слово $\mathbf{W_1}=\{W_{1I} \ldots W_{11}\}$, а последним - слово $\mathbf{W_J}=\{W_{JI} \ldots W_{J1}\}$. Во всех этих словах количество весовых коэффициентов и границы между ними совпадают и определяются содержимым регистра NB1. Последовательность загрузки и расположение весовых коэффициентов в WBUF иллюстрируется на Рисунок 4.9.

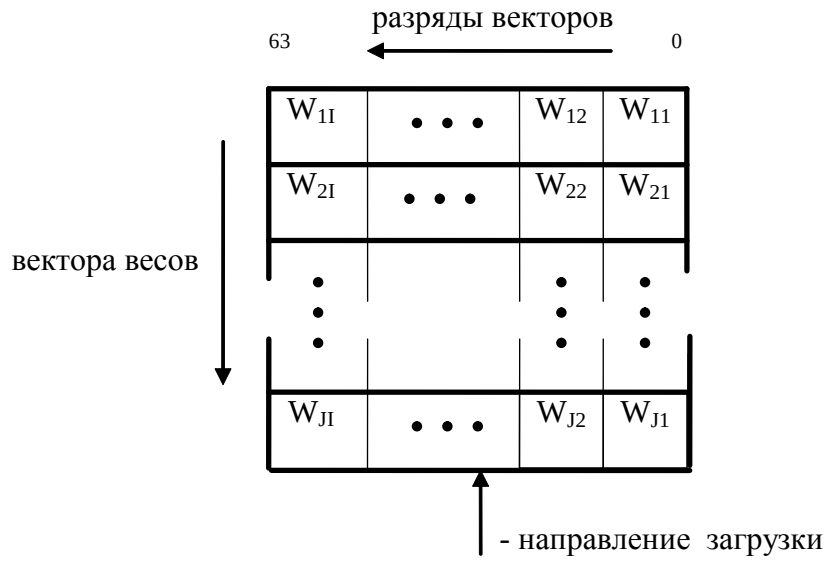


Рисунок 4.9 - Формат матрицы весов WBUF

4.8 FIFO весовых коэффициентов (WFIFO)

Двухпортовое WFIFO имеет емкость 32x64 бит и используется в качестве накопительного буфера в процессе подкачки матрицы весов в WBUF из внешней памяти. Запись и чтение из WFIFO ведется 64-разрядными словами упакованных весовых коэффициентов. Загрузка WFIFO осуществляется через входную шину данных процессорного ядра – WB<63:0>.

WFIFO введено в схему сопроцессора с целью повышения эффективности использования шин процессорного ядра в процессе подкачки весовых коэффициентов в WBUF из внешней памяти. Загрузка матрицы весов в WBUF выполняется от 1 до 32 тактов. При этом в зависимости от содержимого теневого регистра SB1 количество загружаемых слов J может принимать целочисленное значение в диапазоне от 1 до 32. В отсутствии WFIFO непосредственная загрузка матрицы весов из внешней памяти в WBUF, проводимая при малых значениях J, приводила бы к значительным простоям в работе одной из шин. Поэтому процесс загрузки матрицы весов из внешней памяти в WBUF выполняется процессором в два этапа. Первоначально, по команде загрузки WFIFO в него записывается массив из N 64-разрядных слов упакованных весовых коэффициентов, выбираемых из внешней памяти. Значение N задается кодом команды и находится в диапазоне от 1 до 32. Затем по команде загрузки матрицы весов в WBUF J 64-разрядных слов упакованных весовых коэффициентов переписываются из WFIFO в WBUF. Следует отметить, что нецелесообразно использовать значения N, меньшие, чем значения J, так как это может привести к зависанию процессорного ядра. При N>>J в WFIFO можно записать сразу несколько матриц весов, которые затем будут последовательно переписываться в WBUF, причём команды загрузки WFIFO и WBUF могут существенно отстоять друг от друга в общем потоке команд.

Таким образом, WFIFO служит для согласования асинхронных процессов загрузки матрицы весов в WBUF и чтения слов упакованных весовых коэффициентов из внешней памяти. Контроль за данными процессами может осуществляться программно путем анализа флагов “WFIFO пустое” и “WFIFO полное”, которые фиксируются в регистре INTR RISC-ядра.

4.9 Накопительное FIFO (AFIFO)

Двухпортовое AFIFO емкостью 32x64 бита используется в VU в качестве аккумулятора и служит для хранения одного вектора 64-разрядных слов упакованных данных, которые являются результатом выполнения последней векторной операционной команды. В зависимости от кода команды содержимое AFIFO может пересылаться в следующие приемники:

- Во внешнюю память через шину выходных данных (VDOB<63:0>);
- Одновременно во внешнюю память и в VRAM через шину выходных данных;
- На входы исполнительных узлов VU по цепи обратной связи, проходящей через коммутатор 3 в 2;
- Одновременно во внешнюю память и на входы исполнительных узлов VU по цепи обратной связи;
- Одновременно во внешнюю память, в VRAM и на входы исполнительных узлов VU.

Выходы AFIFO и цепь обратной связи VU отделены от шины выходных данных процессорного ядра буфером. Поэтому при пересылке содержимого AFIFO только на входы исполнительных узлов сопроцессора шина выходных данных процессорного ядра остается свободной.

Необходимо отметить, что AFIFO предназначено для хранения только одного вектора слов упакованных данных. Если в последовательности выполняемых команд между двумя векторными операционными командами не встречается ни одной команды чтения из AFIFO и если вторая векторная операционная команда также не требует чтения из AFIFO, то процессор отработает эту команду как команду NOP, и сформирует при этом внутреннее прерывание по неправильной последовательности команд.

Контроль за содержимым AFIFO может осуществляться программно путем анализа флагов “AFIFO пустое” и “AFIFO полное”, которые фиксируются в регистре INTR.

4.10 Векторный регистр VRAM

VRAM представляет собой двухпортовую память типа FIFO емкостью 32x64 бита, которая подключена к шинам векторных данных VU – VRB<63:0>, VDIB<63:0>, VDOB<63:0>. Основное отличие VRAM от обычного FIFO заключается в том, что после чтения из VRAM его содержимое не изменяется. VRAM служит для хранения одного вектора 64-разрядных слов упакованных данных, которые могут передаваться на исполнительные узлы VU через коммутатор 3 в 2. В зависимости от кода команды слова, записываемые в VRAM, выбираются из внешней памяти или из AFIFO.

Необходимо отметить, что VRAM предназначен для хранения только одного вектора слов упакованных данных, поэтому любая запись в RAM приводит к потере ее прежнего содержимого. Вместе с тем, допускается одновременная запись нового пакета данных и чтение предыдущего пакета.

4.11 Коммутатор 3 в 2

Коммутатор 3 в 2 обеспечивает выбор двух источников векторных данных, поступающих на входы исполнительных узлов сопроцессора. В зависимости от кода команды через коммутатор 3 в 2 на вход каждого нелинейного преобразователя могут поступать следующие вектора слов упакованных данных:

- Вектора данных с нулевыми значениями всех разрядов;
- Содержимое VRAM;
- Содержимое AFIFO по цепи обратной связи сопроцессора;

- Вектора данных из внешней памяти по шине VDIB<63:0>.

В случае, когда вектор слов упакованных данных выбирается из внешней памяти, то количество слов в нем задается кодом команды. Если же вектор выбирается из VRAM или AFIFO, то количество слов в нем равно числу 64-разрядных слов, записанных в эти памяти по предыдущим командам. Последнее связано с тем, что содержимое VRAM или AFIFO всегда считывается целиком.

Кроме функций коммутации пакетов векторных данных коммутатор 3 в 2 может выполнять функции поразрядного маскирования векторных данных, пропускаемых на исполнительные узлы VU. Если код векторной команды задает операцию маскирования, то на вход NLT1 поступает результат поразрядной логической операции И над пропускаемым на NLT1 словом упакованных данных и 64-разрядной маской, а на вход NLT2 - результат поразрядной логической операции И над пропускаемым на NLT2 словом упакованных данных и инверсией 64-разрядной маски. При этом векторам слов упакованных данных соответствует вектор масок, который выбирается одновременно с векторами слов упакованных данных из тех же источников и по тем же правилам. Фактически коммутатор 3 в 2 является полным коммутатором 3 в 3, у которого третьим выходом является маска.

Код команды может задавать одновременную обработку до трех векторов упакованных данных. Если при этом выбираются пакеты с разным количеством слов, то процессор отработает эту команду, как команду NOP, и сформирует внутреннее прерывание по запрещенной векторной команде.

4.12 Регистр порогов VR

Регистр порогов VR служит для хранения 64-разряного слова упакованных порогов или смещений. Существуют команды, по которым при выполнении операции взвешенного суммирования содержимое VR подается на вход Y OU.

5 Методы адресации памяти

5.1 Методы адресации команд

При выборе очередной команды из внешней памяти её адрес может вычисляться следующими способами:

- По содержимому счётчика адреса команд с его инкрементацией на 2 (выборка команд идёт по 64 разряда) на линейных участках программы;
- По содержимому AR-регистра или GR-регистра для команд перехода/перехода к подпрограмме с соответствующей модификацией счётчика адреса команд (PC):

PC <- ARi, i = 0, …, 7;
PC <- ARi + GRi, i = 0, …, 7;
PC <- GRi, i = 0, …, 7;
PC <- PC + GRi;

- По содержимому AR-регистра или PC с непосредственно заданным в команде смещением в виде 32-разрядной константы (Const32) для команд перехода/перехода к подпрограмме со смещением с соответствующей модификацией PC:

PC <- ARi + Const32;
PC <- Const32;
PC <- PC + Const32.

- По содержимому стека адресов возврата, находящемуся во внешней памяти и адресуемому по содержимому указателя стека (SP) с его декрементацией на 2 для команд возврата из подпрограммы/прерывания с соответствующей модификацией PC:

PC <- [- - SP].

5.2 Методы адресации скалярных данных

Процессорное ядро поддерживает обмен данными с внешней памятью как скалярными, так и векторными данными, задаваемый соответственно скалярными или векторными командами. Скалярные команды задают единичную пересылку регистр <- память или память <- регистр. В зависимости от кода регистра (т. е. от его разрядности) обмен с памятью осуществляется по 32 или 64 разряда, при этом исполнительный адрес памяти (Address) может вычисляться следующими способами:

- По содержимому AR-регистра или GR-регистра с соответствующей модификацией AR-регистра:

Address <- GRi, i = 0, …,7; нет модификации адресных регистров.
Address <- ARi + GRi; ARi <- ARi + GRi, i = 0, …,7.
Address <- GRi; ARi <- GRi, i = 0, …,7.
Address <- ARi + GRi; i = 0, …,7; нет модификации адресных регистров.
Address <- ARi, i = 0, …,7; нет модификации адресных регистров.
Address <- ARi; ARi <- ARi + GRi, i = 0, …,7.
Address <- ARi - a; ARi <- ARi - a, i = 0, …,7; a = 1 при адресации 32- разрядных данных, a = 2 при адресации 64- разрядных данных.
Address <- ARi; ARi <- ARi + a, i = 0, …,7; a = 1 при адресации 32- разрядных данных, a = 2 при адресации 64- разрядных данных.

- По содержимому AR-регистра с непосредственно заданным в команде смещением в виде 32-разрядной константы (Const32):

Address <- Const32; нет модификации адресных регистров.
Address <- ARi + Const32; ARi <- ARi + Const32, i = 0, …,7.
Address <- Const32; ARi <- Const32, i = 0, …,7.
Address <- ARi + Const32; i = 0, …,7; нет модификации адресных регистров.

5.3 Методы адресации векторных данных

Векторные команды задают обмен с внешней памятью блоками по N 64-разрядных слов, где N определяется кодом команды и может меняться в диапазоне от 1 до 32. Исполнительный адрес памяти при этом вычисляется по тем же правилам, что и для скалярных команд, но N раз, и N раз будет модифицирован соответствующий адресный регистр. Отличие состоит лишь в том, что для векторных команд нельзя использовать адресацию с помощью непосредственно заданного в команде смещения в виде 32-разрядной константы. Все остальные методы адресации по содержимому AR-регистра или GR-регистра с соответствующей модификацией AR-регистра остаются доступными:

- Address <- GRi, i = 0, …,7; нет модификации адресных регистров.
  Address <- ARi + GRi; ARi <- ARi + GRi, i = 0, …,7.
  Address <- GRi; ARi <- GRi, i = 0, …,7.
- Address <- ARi + GRi; i = 0, …,7; нет модификации адресных регистров.
- Address <- ARi, i = 0, …,7; нет модификации адресных регистров.
  Address <- ARi; ARi <- ARi + GRi, i = 0, …,7.
  Address <- ARi - 2; ARi <- ARi - 2, i = 0, …,7.
  Address <- ARi; ARi <- ARi + 2, i = 0, …,7.

5.4 Особенности работы с системным стеком и стеками пользователя

Процессорное ядро имеет специальный регистр-указатель стека SP, с помощью которого организуется стек во внешней памяти. Этот стек используется в качестве системного при вызовах подпрограмм, обработке прерываний или возврате из них. Пользователь имеет возможность использовать его для передачи параметров между процедурами или для временного хранения своих данных, а также организовать свой стек, используя в качестве указателя стека любой из адресных регистров - AR0 - AR6.

5.4.1 Системный стек

Указатель стека SP - это 32-разрядный регистр, который содержит адрес вершины системного стека. Стек наполняется от младших адресов к старшим. SP всегда указывает на адрес, по которому будет записан следующий элемент (см. Рисунок 5.1). Запись в стек идёт в режиме постинкремента, чтение - в режиме декремента.

Запись в стек осуществляется каждый раз, когда выполняется команда перехода к подпрограмме или осуществляется вход в прерывание. В этом случае в память идёт запись по адресу в SP 64-разрядного слова, старшая часть которого содержит 32-разрядное слово состояния процессора, а младшая - 32-разрядный счётчик адреса команд, а сам SP затем увеличивается на два.

Рисунок 5.1 - Конфигурация системного стека

Чтение из стека происходит каждый раз, когда встречается команда возврата из подпрограммы или прерывания. В данном случае содержимое SP уменьшается на 2, по этому адресу из стека читается и восстанавливается счётчик адреса команд и старое слово состояния процессора (только для возврата из прерывания).

Вышеописанный механизм позволяет легко создавать программные прерывания, когда при вызове обработчика прерывания можно использовать обычную команду перехода к подпрограмме, а выйти из него можно командой возврата из прерывания.

Пользователь может также писать и читать из системного стека, пользуясь обычными методами адресации относительно AR7(SP), но при этом он должен строго следить за тем, чтобы содержимое SP было чётным. В противном случае при переходе к подпрограммам или при обработке прерываний часть данных в стеке может быть потеряна.

После системного сброса содержимое SP не определено, поэтому необходимо позаботиться о записи в него адреса начала системного стека до того, как будут разрешены прерывания или встретится хоть одна команда перехода к подпрограмме. Не существует ограничений на расположение системного стека во внешней или во внутренней памяти, лишь бы адрес начала стека был выровнен по границе 64-разрядного слова.

5.4.2 Аппаратная вершина системного стека

Для ускорения выполнения команд возврата из подпрограммы/прерывания в микросхеме была введена аппаратная вершина стека. Программно она может отключаться с помощью обнуления 12-го разряда регистра PSWR, и тогда работа с системным стеком происходит так, как описано в предыдущем разделе. После системного сброса аппаратная вершина стека также отключена.

Чтобы включить аппаратную вершину стека, необходимо записать в 12-й разряд регистра PSWR единицу. После этого первая же команда перехода к подпрограмме или первый же вход в прерывание дублируют запись адреса возврата и PSWR в системный стек, находящийся во внешней или внутренней памяти, и в аппаратную вершину стека. При втором и дальнейших переходах к подпрограмме/входах в прерывание, в аппаратной вершине стека помимо последнего адреса возврата и копии PSWR добавляется информация, позволяющая вычислить адрес предыдущей записи в системном стеке по команде перехода к подпрограмме/ по входу в прерывание.

При выполнении команды возврата из подпрограммы/прерывания адрес возврата и состояние PSWR считывается из аппаратной вершины стека, а в саму вершину из системного стека подкачивается новое значение, позволяющее выполнить следующую команду возврата

из подпрограммы/прерывания. Тем самым поддерживаются многоуровневые входы/выходы в подпрограммы/прерывания практически неограниченной глубины.

5.4.3 Стеки пользователя

Пользователь имеет возможность сам создать свой стек, используя в качестве указателя стека любой из адресных регистров - AR0 - AR6. Обмен со стеком будет осуществляться с помощью обычных методов адресации: запись в режиме постинкремента, чтение - в режиме декремента. В данном случае можно писать и читать из стека как 32-разрядные, так и 64-разрядные данные, причём в первом случае указатель стека будет меняться на +/- 1, во втором на +/- 2. Ограничение существует только на обмен 64-разрядными данными: указатель стека должен быть в этот момент выровнен на границу 64-разрядного слова. В остальном работа со стеком пользователя аналогична работе с системным стеком.

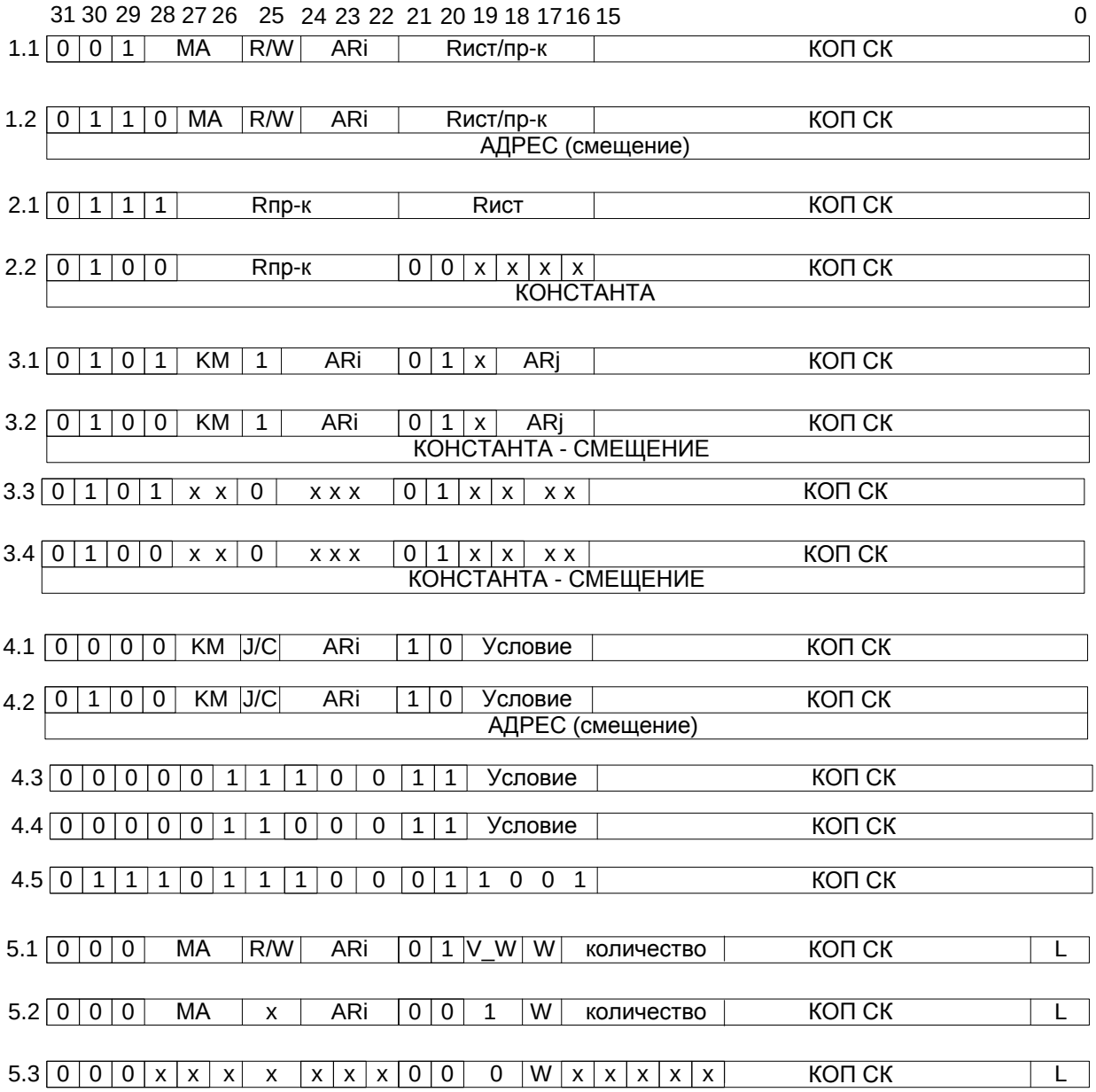
6 Введение в систему команд

Система команд микросхемы была разработана и оптимизирована так, чтобы обеспечить ему максимальную эффективность при решении задач обработки больших потоков данных. Система команд микросхемы совместима с системой команд других процессоров этого же семейства и является уникальной по отношению к другим процессорам.

Форматы команд приведены на рисунке 6.1, краткое их описание дано ниже.

Характерными особенностями системы команд микросхемы являются:

- Фиксированная длина команды. Все команды имеют длину 32 разряда либо 64 разряда, если старшим словом команды является непосредственно передаваемая константа;



а) Команды управляющего RISC-процессора

Рисунок 6.1(а) - Система команд процессорного ядра NMC4

- фиксированная длина команды. Все команды имеют длину 32 разряда либо 64 разряда, если старшим словом команды является непосредственно передаваемая константа;
- сверхбольшое слово команды (VLIW). Каждая команда может задавать несколько операций, причем эти операции могут исполняться независимо;
- аппаратная поддержка циклов. В командах, определяющих работу матрично-векторного сопроцессора, допускается задавать число повторений команды от 1 до 32;
- принудительное включение последовательного исполнения команд. Процессорное ядро является мультиконвейерным, динамическим суперскалярным вычислительным устройством, которое может исполнять несколько команд одновременно.

Команды процессорного ядра NMC4 делятся на две основные группы: команды управляющего RISC-процессора (см. Рисунок 6.1 а) и команды сопроцессора арифметики с плавающей точкой (см. Рисунок 6.1 б). Каждая из этих групп, в свою очередь, делится на команды скалярные, т. е. обычные RISC-команды, и векторные, которые задают многократно одни и те же действия над векторами данных, что эквивалентно аппаратной поддержке коротких циклов. В отличие от векторных команд сопроцессора арифметики с плавающей точкой, векторные команды управляющего RISC-процессора поддерживают только операции над данными, представленными в формате с фиксированной точкой.

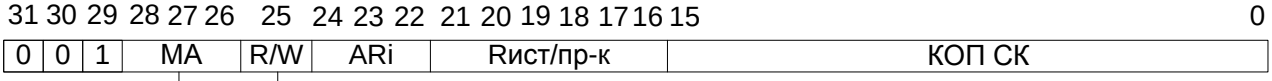
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	…	0
1.1F	1	0	1	MA			R/W	ARi			Rист/пр-к							КОП СК	
1.2F	1	1	1	0	MA		R/W	ARi			Rист/пр-к							КОП СК	
	АДРЕС (смещение)																		
2.1F	1	1	1	1	Rпр-к						Rист							КОП СК	
2.2F	1	1	0	0	Rпр-к						0	0	x	x	x	x		КОП СК	
	КОНСТАНТА																		
3.1F	1	0	0	M		0	0	OP			0 0 0 0		DT	x	VR_A	x x x x x x x		VRD	CELL_D
3.2F	1	0	0	M		0	1	x	OP		0 0 0 0		DT	x	x x	SB	VR_B	VR_C	VRD	CELL_D
3.3F	1	0	0	M		1	0	OP			0 0 0 0		DT	WE	VR_A	SB	VR_B	MGC	VRD	CELL_D
3.4F	1	0	0	M		1	1	OP			0 0 0 0		DT	SA	VR_A	SB	VR_B	VR_C	VRD	CELL_D
4.1F	1	0	0	MA		0		ARi			0	0	1	x	количество	R	x x x x x x		VRD	CELL_D
4.2F	1	0	0	MA		1		ARi			0	0	1	x	количество	R	VRS	CELL_S	x x x x x x	
4.3F	1	0	0	MA		0		ARi			0	1	0	x	количество	R	x x x x x x		RPOP	
4.4F	1	0	0	MA		1		ARi			0	1	0	x	количество	R	x x x x x x x x x x x x			
5.1F	1	0	0	x x x x			0	0	1 0 0 0		x x x x x x x					VRS	CELL_S	VRD	CELL_D	
5.2F	1	0	0	x x x x			0	1	1 0 0 0		x x x			VRS0	MS	VRS1	CELL_S	VRD	CELL_D	
5.3F	1	0	0	x x x x			1	0	1 0 0 0		x x x x x x				PR	VRS1	CELL_S	VRD	CELL_D	
5.4F	1	0	0	x x x x			1	1	1 0 0 0		x			количество	R	x x x x x		PR	VRD	CELL_D

б) Команды сопроцессора арифметики с плавающей точкой

Рисунок 6.1(б) - Система команд процессорного ядра NMC4 (FPU)

6.1 Форматы команд, задающих пересылку данных типа «регистр управляющего RISC-процессора – память»

Формат 1.1 Rист/пр- к ↔(fадр.(ARi, GRi)); ARi←fм(ARi, GRi)

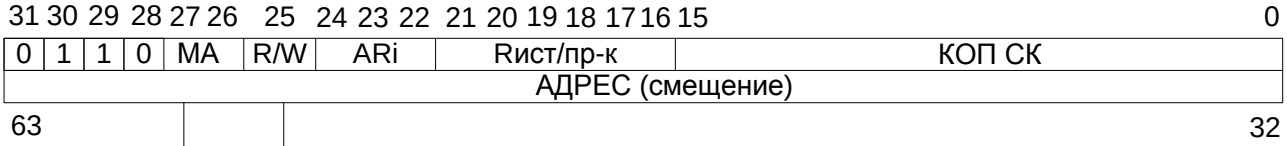


R/W	Направление пересылки данных
0	Чтение из памяти в Rист/пр-к
1	Запись в память из Rист/пр-к

MA			$f_{адр.}(AR_i, GR_i)$	$f_м(AR_i, GR_i)$
0	0	0	GR_i	AR_i
0	0	1	AR_i+GR_i	AR_i+GR_i
0	1	0	GR_i	GR_i
0	1	1	AR_i+GR_i	AR_i
1	0	0	AR_i	AR_i
1	0	1	AR_i	AR_i+GR_i
1	1	0	AR_i- a	AR_i- a
1	1	1	AR_i	AR_i+a

Примечание: a=1 при адресации 32- разрядных данных;
a=2 при адресации 64- разрядных данных.

Формат 1.2 Rист/пр- к ↔(fадр.(ARi, CM)); ARi←fм(ARi, CM)



R/W	Направление пересылки данных
0	Чтение из памяти в Rист/пр-к
1	Запись в память из Rист/пр-к

MA		$f_{адр.}(AR_i, CM)$	$f_м(AR_i, CM)$
0	0	GR_i	AR_i
0	1	AR_i+CM	AR_i+CM
1	0	Адрес	Адрес
1	1	AR_i+CM	AR_i

6.2 Форматы команд, задающих пересылку данных типа “регистр-регистр» между регистрами управляющего RISC-процессора

Формат 2.1 Rист → Rпр-к

31	30	29	28	27 26	25 24 23 22	21 20 19 18 17 16 15	0
0	1	1	1	Rпр-к	Rист	КОП СК	

Формат 2.2 Rпр-к ← КОНСТАНТА

31	30	29	28	27 26	25 24 23 22	21	20	19	18	17	16	15	0
0	1	0	0	Rпр-к		0	0	x	x	x	x	КОП СК	
КОНСТАНТА													

63 32

6.3 Форматы команд модификации адресных регистров управляющего RISC-процессора

Формат 3.1. ARj←fм(ARi,GRi)

31	30	29	28	27 26	25	24 23 22	21	20	19	18 17 16 15	0
0	1	0	1	KM	1	ARi	0	1	x	ARj	КОП СК

KM		$fм(AR_i, GR_i)$
0	0	ARi
0	1	ARi+GRi
1	0	GRi
1	1	PC+GRi

ARпр-к			AR приёмник
0	0	0	AR0
0	0	1	AR1
0	1	0	AR2
0	1	1	AR3
1	0	0	AR4
1	0	1	AR5
1	1	0	AR6
1	1	1	SP(AR7)

Формат 3.2 ARj←fм(ARi, CM)

31	30	29	28	27 26	25	24 23 22	21	20	19	18 17 16 15	0
0	1	0	0	KM	1	ARi	0	1	x	ARj	КОП СК
КОНСТАНТА-СМЕЩЕНИЕ											

63 32

KM		fм(ARi, CM)
0	0	ARi
0	1	ARi+CM
1	0	CM
1	1	PC+CM

ARпр-к	См. формат 3.1

Формат 3.3. Нет операций ввода/вывода и модификации адресных регистров

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	0
0	1	0	1	x	x	0	x	x	x	0	1	x	x	x	x	КОП СК	

Формат 3.4 Нет операций ввода/вывода и модификации адресных регистров

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	0
0	1	0	0	KM		1	ARi			0	1	x	ARj			КОП СК	
КОНСТАНТА-СМЕЩЕНИЕ																	
63																	32

6.4 Форматы команд управления RISC-процессора

Формат 4.1 Переход/переход к подпрограмме

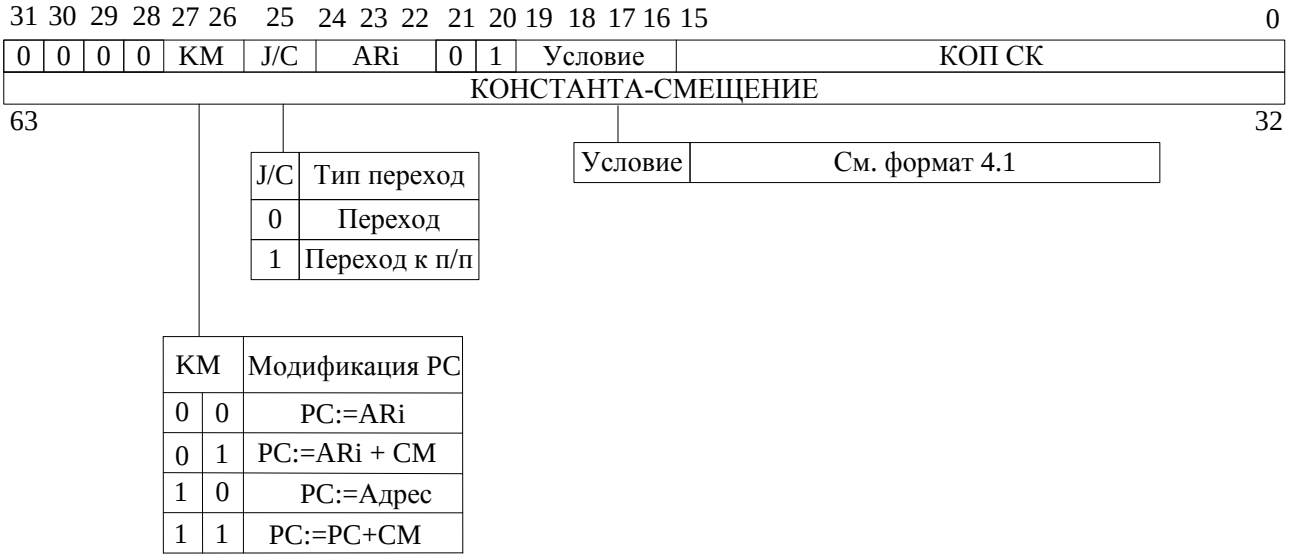
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	0
0	0	0	0	KM		J/C	ARi			0	1	Условие				КОП СК	

J/C	Тип переход
0	Переход
1	Переход к п/п

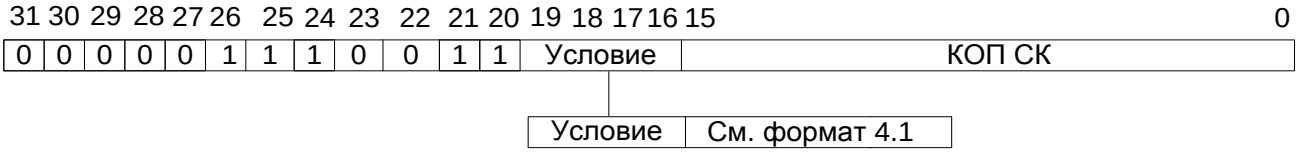
KM		Модификация PC
0	0	PC:=ARi
0	1	PC:=ARi + GRi
1	0	PC:=GRi
1	1	PC:=PC+GRi

Условие				Анализируемое сочетание признаков
0	0	0	0	Переход, если C=0
0	0	0	1	Переход, если V=0
0	0	1	0	Переход, если N+Z=0
0	0	1	1	Переход, если N=0
0	1	0	0	Переход, если (V⊕N)+Z=0
0	1	0	1	Переход, если V⊕N=0
0	1	1	0	Переход, если Z=0
0	1	1	1	Переход безусловный
1	0	0	0	Переход, если C=1
1	0	0	1	Переход, если V=1
1	0	1	0	Переход, если N+Z=1
1	0	1	1	Переход, если N=1
1	1	0	0	Переход, если (V⊕N)+Z=0
1	1	0	1	Переход, если V⊕N=1
1	1	1	0	Переход, если Z=1
1	1	1	1	Перехода нет

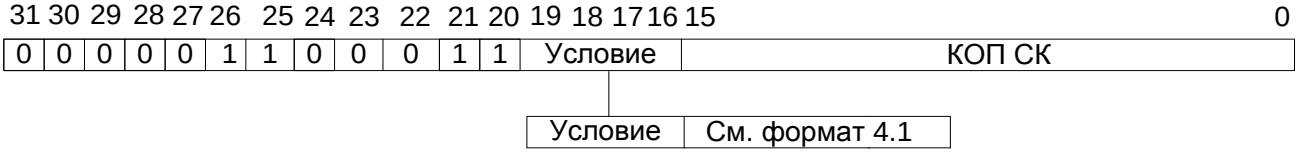
Формат 4.2. Переход/переход к подпрограмме со смещением



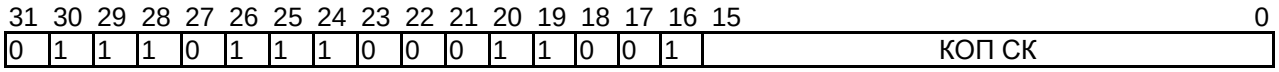
Формат 4.3. Возврат из подпрограммы



Формат 4.4. Возврат из прерывания

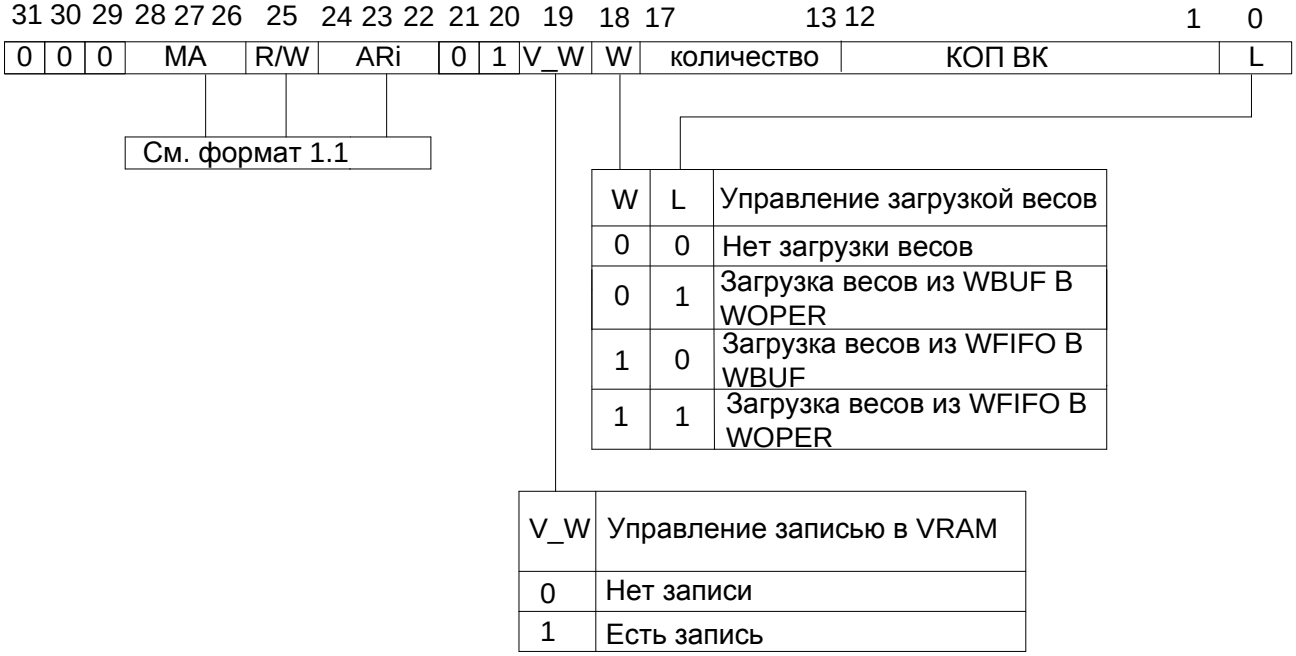


Формат 4.5. Останов (HALT)



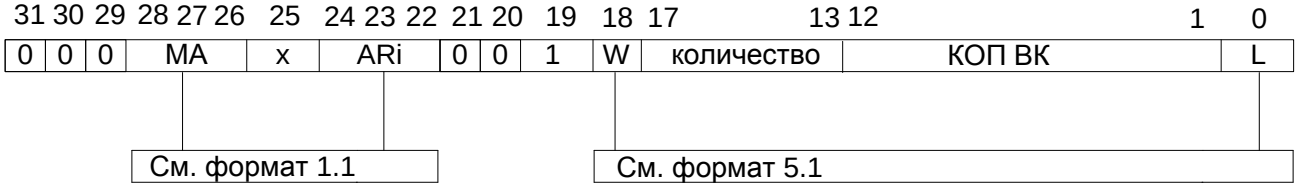
6.5 Форматы команд управления RISC-процессора

Формат 5.1. Векторные данные ↔ (fадр.(ARi, GRi)); ARi←fм(ARi, GRi)



Формат 5.2. Загрузка весов в WFIFO из внешней памяти

WFIFO←(fадр.(ARi, GRi)); ARi←fм(ARi, GRi)

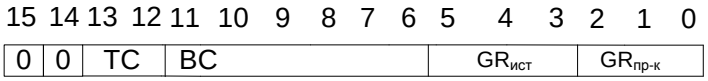


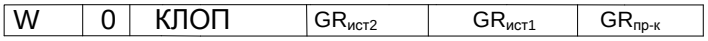
Формат 5.3. Нет операций ввода/вывода

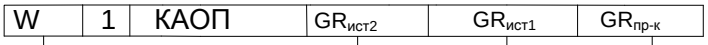
Примечание: 1)поле “Количество” задает количество 64-разрядных слов, участвующих в операциях пересылок данных:

00000 - одно 64-разрядное слово;
00001 - два 64-разрядных слова;
…
11111 – тридцать два 64-разрядных слова.

6.6 Форматы поля КОП СК (код операции скалярной команды управляющего RISC-процессора)



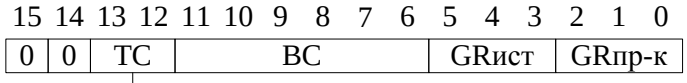




GR			РОН – источник/приемник
0	0	0	GR0
0	0	1	GR1
0	1	0	GR2
0	1	1	GR3
1	0	0	GR4
1	0	1	GR5
1	1	0	GR6
1	1	1	GR7

W		Управление записью в $GR_{пр-к}$ и в регистр признаков
0	1	Есть запись в $GR_{пр-к}$, нет записи признаков
1	0	Есть запись признаков, нет записи в $GR_{пр-к}$
1	1	Есть запись в $GR_{пр-к}$ и есть запись признаков

6.7 Формат поля КОП СК, задающего операцию сдвига



ТС		Тип сдвига
0	0	Циклический сдвиг
0	1	Логический сдвиг
1	0	Арифметический сдвиг
1	1	Логический сдвиг через «С»

Примечания

1 Поле ВС задает величину сдвига в дополнительном коде:
- положительное значение значение - сдвиг влево (от 0 до 31 разряда),
- отрицательное значение - сдвиг вправо (от 1 до 32 разрядов);

2 “С” - признак переноса слова состояния процессора;

3 Логический сдвиг через “С” может выполняться только на один разряд влево или вправо; сдвиги остальных типов могут выполняться на любое от 1 до 31 число разрядов влево или вправо;

4 Сдвиг на 0 или 32 разряда любого типа воспринимается как код “нет операции”, и при этом не изменяется ни приёмник результата операции GRпр-к, ни признаки;

5 Сдвиги от 1 до 31 разряда меняют признаки слова состояния процессора по следующим правилам:

- N - признак знака - равен старшему (знаковому) разряду результата;
- Z - признак нуля - равен единице, если все разряды регистра приемника GR нулевые, или нулю в противном случае;
- C - признак переноса, равен последнему выталкиваемому при сдвиге разряду из GR источника.

Схемы различных вариантов сдвига на один разряд изображены на Рисунок 6.2. Сдвиги на большее число разрядов эквивалентны многократному сдвигу на единицу, хотя и выполняются за один такт работы процессора.

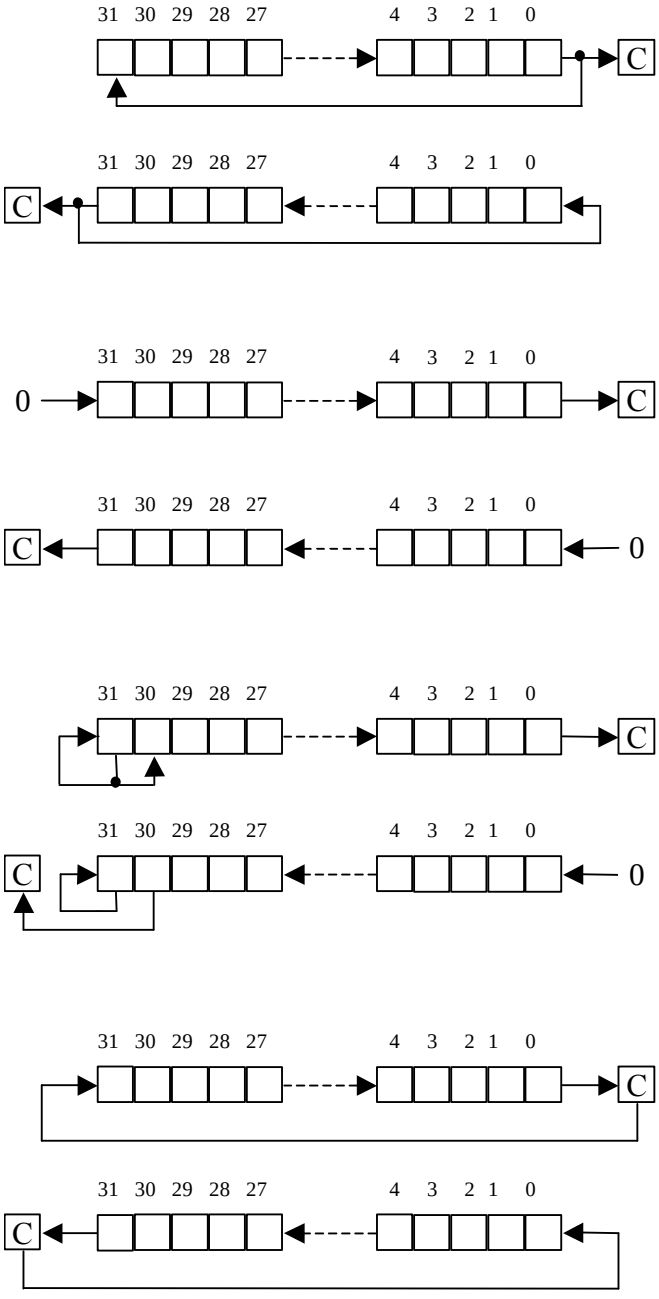


Рисунок 6.2 - Схемы выполнения различных типов операций сдвигов

6.8 Формат поля КОП СК, задающего логическую операцию

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W		0	КЛОП				$GR_{ист2}$			$GR_{ист1}$			GRпр-к		

КЛОП				Код логической операции	N	Z	V	C
0	0	0	0	0	0	1	0	0
0	0	0	1	$\overline{GR}_{ист2}$ & $\overline{GR}_{ист1}$	+	+	0	0
0	0	1	0	$GR_{ист2}$ & $\overline{GR}_{ист1}$	+	+	0	0
0	0	1	1	$\overline{GR}_{ист1}$	+	+	0	0
0	1	0	0	$\overline{GR}_{ист2}$ & $GR_{ист1}$	+	+	0	0
0	1	0	1	$\overline{GR}_{ист2}$	+	+	0	0
0	1	1	0	$GR_{ист2} \oplus GR_{ист1}$	+	+	0	0
0	1	1	1	$\overline{GR}_{ист2}$ + $\overline{GR}_{ист1}$	+	+	0	0
1	0	0	0	$GR_{ист2}$ & $GR_{ист1}$	+	+	0	0
1	0	0	1	$GR_{ист2} \oplus \overline{GR}_{ист1}$	+	+	0	0
1	0	1	0	$GR_{ист2}$	+	+	0	0
1	0	1	1	$GR_{ист2}$ + $\overline{GR}_{ист1}$	+	+	0	0
1	1	0	0	$GR_{ист1}$	+	+	0	0
1	1	0	1	$\overline{GR}_{ист2}$ + $GR_{ист1}$	+	+	0	0
1	1	1	0	$GR_{ист2}$ + $GR_{ист1}$	+	+	0	0
1	1	1	1	- 1	1	0	0	0

6.9 Формат поля КОП СК, задающего арифметическую операцию

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W		1	КАОП				$GR_{ист2}$			$GR_{ист1}$			$GR_{пр-к}$		

КАОП				Код арифметической операции	Y	N	Z	V	C
0	0	0	0	$GR_{ист2}$ - $GR_{ист1}$	-	+	+	+	+
0	0	0	1	$GR_{ист2}$ - $GR_{ист1}$ – 1 + “C”	-	+	+	+	+
0	0	1	0	$GR_{ист2}$ + 1	-	+	+	+	+
0	0	1	1	$GR_{ист2}$ + “C”	-	+	+	+	+
0	1	0	0	$GR_{ист2}$ -1	-	+	+	+	+
0	1	0	1	$GR_{ист2}$ -1 + “C”	-	+	+	+	+
0	1	1	0	$GR_{ист2}$ + $GR_{ист1}$	-	+	+	+	+
0	1	1	1	$GR_{ист2}$ + $GR_{ист1}$ + “C”	-	+	+	+	+
1	0	0	0	Первый шаг умножения	+	?	?	0	?
1	0	0	1	Шаг умножения	+	?	?	0	?
1	0	1	x	Резерв					
1	1	0	0	-$GR_{ист2}$	-	+	+	+	+
1	1	x	1	Резерв					
1	1	1	x	Резерв					

Примечания

1 C - признак переноса из слова состояния процессора;

2 Если в поле W задаёт запись признаков, то они устанавливаются соответственно столбцам N, Z, V, C, иначе они сохраняют своё значение;

3 Признак Y меняется только операциями “Первый шаг умножения” и “Шаг умножения” независимо от поля W;

4 В таблицах используются следующие обозначения:

1. “+” - признак устанавливается по результату операции;
2. “0” - признак обнуляется;
3. “1” - признак устанавливается в единицу;
4. “?” - значение признака не определено;

5 Признак Y равен предпоследнему по значимости (1- му) разряду множителя (GR7) при выполнении операций “Первый шаг умножения” и “Шаг умножения”;

6 Признак N равен 31 (знаковому) разряду результата;

7 Признак Z равен единице, если все разряды результата нулевые, иначе – нулю;

8 Признак V равен нулю для операций шага умножения, для остальных операций он формируется как результат операции “исключающее ИЛИ” переноса из 31 разряда и переноса в 31 разряд;

9 Признак C равен переносу из 31 разряда.

6.10 Формат поля КОП ВК (код операции векторной команды управляющего RISC-процессора)

12	11	10	9	8	7	6	5	4	3	2	1	
1	0	КЛОП				FPX	FPY	X			Y	Логическая операция над операндами X и У
1	1	КАОП				FSX	FSY	X			Y	
0	1	M	0	SH		0	FPY	X			Y	Операция маскирования $X*M+Y*\overline{M}$
0	1	M	0	0		FPX	FPY	X			Y	
0	1	0	0	1	X	X	X	0	0	0	0	Запись в AFIFO содержимого регистров F2CR,F1CR, NB2, SB2, VR
0	0	0	0	0	X	X	X	0	0	0	0	Нет операции
0	0	M	VR	SH		0	FSY	X			Y	Операция взвешенного суммирования типа:W*X+Y
0	0	M	VR	0		FSX	FSY	X			Y	

OP		Выбор операнда
0	0	Операнд равен нулю
0	1	Операнд из RAM
1	0	Операнд из AFIFO
1	1	Операнд выбирается из внешней памяти

FP(FA)	Управление пороговой функцией (функцией насыщения)
0	Функция не используется
1	Функция используется

6.11 Формат поля КОП ВК, задающего логическую операцию

12	11	10	9	8	7	6	5	4	3	2	1
1	0	КЛОП				FPX	FPY	X		Y	

КЛОП				Код логической операции
0	0	0	0	0
0	0	0	1	$\overline{X}$ & $\overline{Y}$
0	0	1	0	X & $\overline{Y}$
0	0	1	1	$\overline{Y}$
0	1	0	0	$\overline{X}$ & Y
0	1	0	1	$\overline{X}$
0	1	1	0	X⊕Y
0	1	1	1	$\overline{X}$ + $\overline{Y}$
1	0	0	0	X & Y
1	0	0	1	$\overline{X}$⊕Y
1	0	1	0	X
1	0	1	1	X + $\overline{Y}$
1	1	0	0	Y
1	1	0	1	$\overline{X}$ + Y
1	1	1	0	X + Y
1	1	1	1	- 1

6.12 Формат поля команд КОП ВК, задающего арифметическую операцию

12	11	10	9	8	7	6	5	4	3	2	1
1	1	КАОП				FSX	FSY	X		Y	

КАОП				Код логической операции
x	0	0	x	X - Y
x	0	1	x	X + 1
x	1	0	x	X - 1
x	1	1	x	X + Y

6.13 Операции маскирования и взвешенного суммирования

12	11	10 9	8	7	6	5	4 3	2 1
0	1	M	0	0	FPX	FPY	X	Y
0	1	M	0	SH	0	FPY	X	Y

Операция маскирования X*M+Y*$\overline{M}$

0	0	M	VR	0	FSX	FSY	X	Y
0	0	M	VR	SH	0	FSY	X	Y

Операция взвешенного суммирования типа W*X+Y

SH	Управление циклическим сдвигом операнда X на разряд вправо
0	Нет сдвига
1	Есть сдвиг

VR	Управление выборкой регистра VR в качестве операнда Y
0	VR не является операндом Y
1	VR является операндом Y

M		Выбор операнда - маски
0	0	Маскирование отсутствует
0	1	Маска выбирается из RAM
1	0	Маска выбирается из AFIFO
1	1	Маска выбирается из внешней памяти

6.14 Поле выбора адресного регистра управляющего RISC-процессора

25	24	23	
ARi			Номер AR или GR
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Примечание: AR7(SP) может использоваться в качестве системного указателя стека адресов возврата при входе/выходе из подпрограммы (прерывания), причём изменяется он в этом случае только на +/- 2.

6.15 Поле $R_{ист/пр-к}$ в командах пересылки данных управляющего RISC-ядра

Таблица 6.1 - Поле Rист/пр-к в командах пересылки данных управляющего RISC-ядра

Код регистра	Регистр- источник	Регистр- приемник
0 0 0 0 0 0	GR0,AR0	GR0,AR0
0 0 0 0 0 1	GR1,AR1	GR1,AR1
0 0 0 0 1 0	GR2,AR2	GR2,AR2
0 0 0 0 1 1	GR3,AR3	GR3,AR3
0 0 0 1 0 0	GR4,AR4	GR4,AR4
0 0 0 1 0 1	GR5,AR5	GR5,AR5
0 0 0 1 1 0	GR6,AR6	GR6,AR6
0 0 0 1 1 1	GR7,AR7	GR7,AR7
0 0 1 0 0 0	AR0	AR0
0 0 1 0 0 1	AR1	AR1
0 0 1 0 1 0	AR2	AR2
0 0 1 0 1 1	AR3	AR3
0 0 1 1 0 0	AR4	AR4
0 0 1 1 0 1	AR5	AR5
0 0 1 1 1 0	AR6	AR6
0 0 1 1 1 1	AR7(SP)	AR7(SP)
0 1 0 0 0 0	GR0	GR0
0 1 0 0 0 1	GR1	GR1
0 1 0 0 1 0	GR2	GR2
0 1 0 0 1 1	GR3	GR3
0 1 0 1 0 0	GR4	GR4
0 1 0 1 0 1	GR5	GR5
0 1 0 1 1 0	GR6	GR6
0 1 0 1 1 1	GR7	GR7
0 1 1 0 0 0	PSWR,PC	PSWR,PC
0 1 1 0 0 1	SIR	SIR
0 1 1 0 1 0	INTR	INTRreset
0 1 1 0 1 1	VL	VL
0 1 1 1 0 0	PC	PC
0 1 1 1 0 1	-	PSWRset
0 1 1 1 1 0	-	PSWRreset
0 1 1 1 1 1	PSWR	PSWR
1 0 0 0 0 0	-	NB
1 0 0 0 0 1	-	SB
1 0 0 0 1 0	-	F1CR
1 0 0 0 1 1	-	F2CR
1 0 0 1 0 0	-	VR
1 0 0 1 0 1	-	-
1 0 0 1 1 0	INTR	-
1 0 0 1 1 1	PC	-
1 0 1 0 0 0	-	NBL
1 0 1 0 0 1	-	SBL
1 0 1 0 1 0	-	F1CRL
1 0 1 0 1 1	-	F2CRL
1 0 1 1 0 0	-	VRL
1 0 1 1 0 1	-	-
1 0 1 1 1 0	-	-
1 0 1 1 1 1	-	-

Продолжение таблицы 6.1

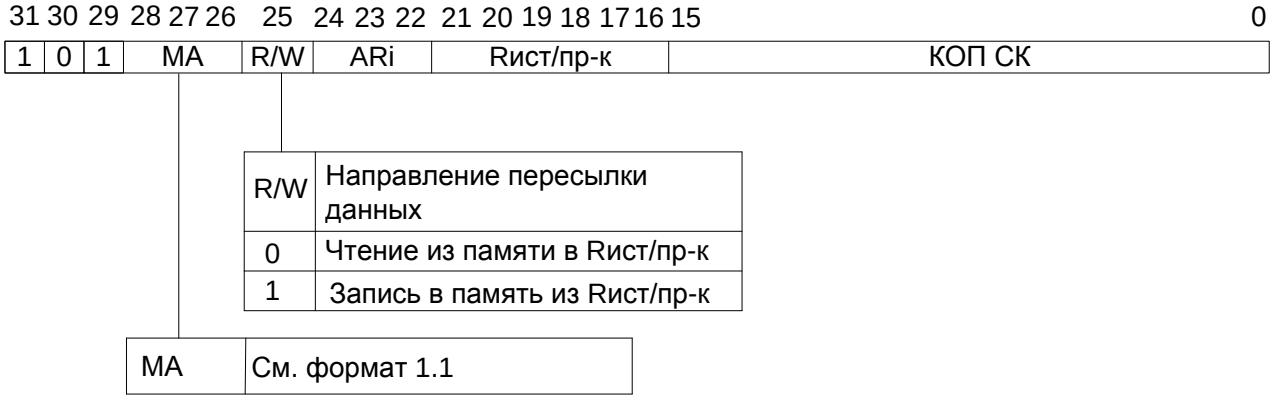
Код регистра	Регистр- источник	Регистр- приемник
1 1 0 0 0 0	-	NBH
1 1 0 0 0 1	-	SBH
1 1 0 0 1 0	-	F1CRH
1 1 0 0 1 1	-	F2CRH
1 1 0 1 0 0	-	VRH
1 1 0 1 0 1	-	-
1 1 0 1 1 0	-	-
1 1 0 1 1 1	-	-
1 1 1 0 0 0	-	-
1 1 1 0 0 1	-	SIR
1 1 1 0 1 0	-	-
1 1 1 0 1 1	-	-
1 1 1 1 0 0	-	-
1 1 1 1 0 1	-	-
1 1 1 1 1 0	-	-
1 1 1 1 1 1	-	-

Таблица 6.2 - Обозначение регистров управляющего RISC-процессора и их назначение

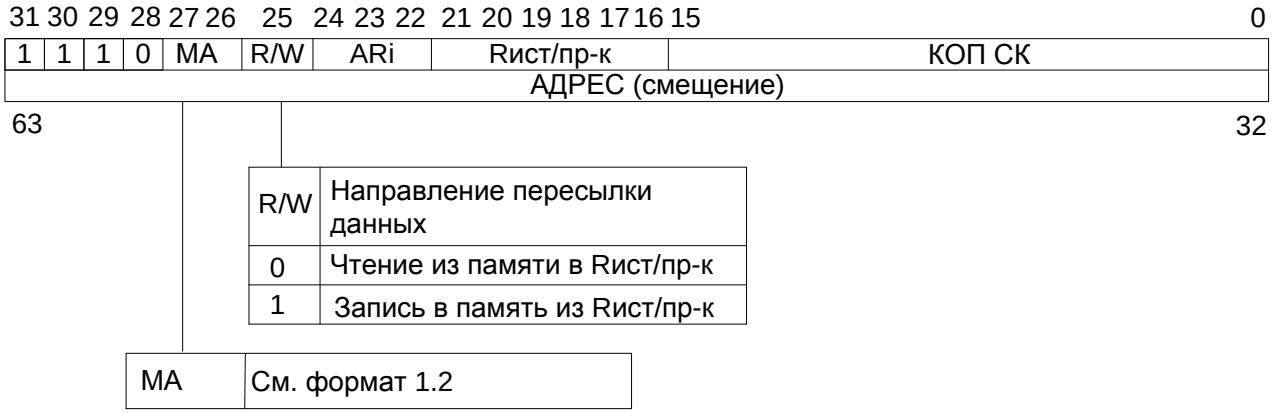
Обозначение регистров и их назначение	Разрядность
GRi – регистр общего назначения I (i=0,…,7)	32
ARj – адресный регистр j (j=0,…,6)	32
SP(AR7) – указатель стека адресов возврата	32
PC – программный счетчик	32
PSWR – регистр слова состояния процессора	32
PSWRset – код для побитовой установки PSWR в единицу (псевдорегистр)	-
PSWRreset – код для побитового сброса PSWR в ноль (псевдорегистр)	-
INTR – регистр запросов на прерывание	32
INTRreset – код для побитового сброса INTR в ноль (псевдорегистр)	-
VL – регистр, задающий длину вектора для векторных команд сопроцессора арифметики с плавающей точкой	32
SIR – регистр системного интегратора, через который осуществляются пересылки между регистрами RISC-процессора и сопроцессоров	64
FiCR (H,L) – регистр управления функцией насыщения i (i=1,2)	64
FiCRH – старшая половина регистра управления функцией насыщения i (i=1,2)	32
FiCRL – младшая половина регистра управления функцией насыщения i (i=1,2)	32
VR – регистр порога	64
VRH – старшая половина регистра порога	32
VRL – младшая половина регистра порога	32
NB – регистр границ нейронов	64
NBH – старшая половина регистра границ нейронов	32
NBL – младшая половина регистра границ нейронов	32
SB – регистр границ синапсов	64
SBH – старшая половина регистра границ синапсов	32
SBL – младшая половина регистра границ синапсов	32

6.16 Форматы команд, задающих пересылку данных типа «регистр сопроцессора арифметики с плавающей точкой - память»

Формат 1.1F Rист/пр- к ↔(fадр.(ARi, GRi)); ARi←fм(ARi, GRi)

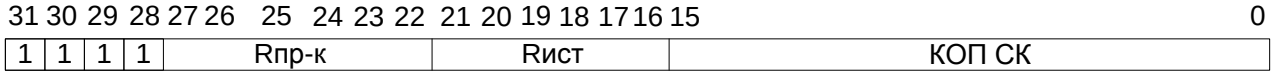


Формат 1.2F Rист/пр- к ↔(fадр.(ARi, СМ)); ARi←fм(ARi, СМ)

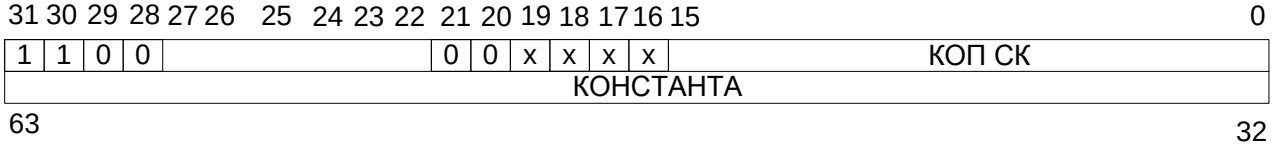


6.17 Форматы команд, задающих пересылку данных типа «регистр - регистр» сопроцессора арифметики с плавающей точкой

Формат 2.1F Rист → Rпр-к

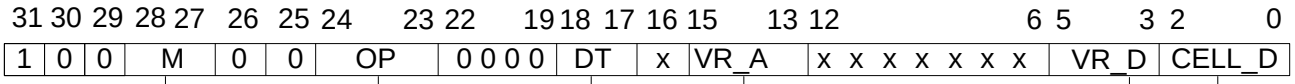


Формат 2.2F РЕЗЕРВ (Команда запрещена)



6.18 Арифметические векторные команды сопроцессора арифметики с плавающей точкой

Формат 3.1F (fap.(VR_A)) → VR_D



Поле маски АЛУ операции	M
Нет маски	0 0
Нет маски	0 1
Маскирование по нулевому значению маски	1 0
Маскирование по единичному значению маски	1 1

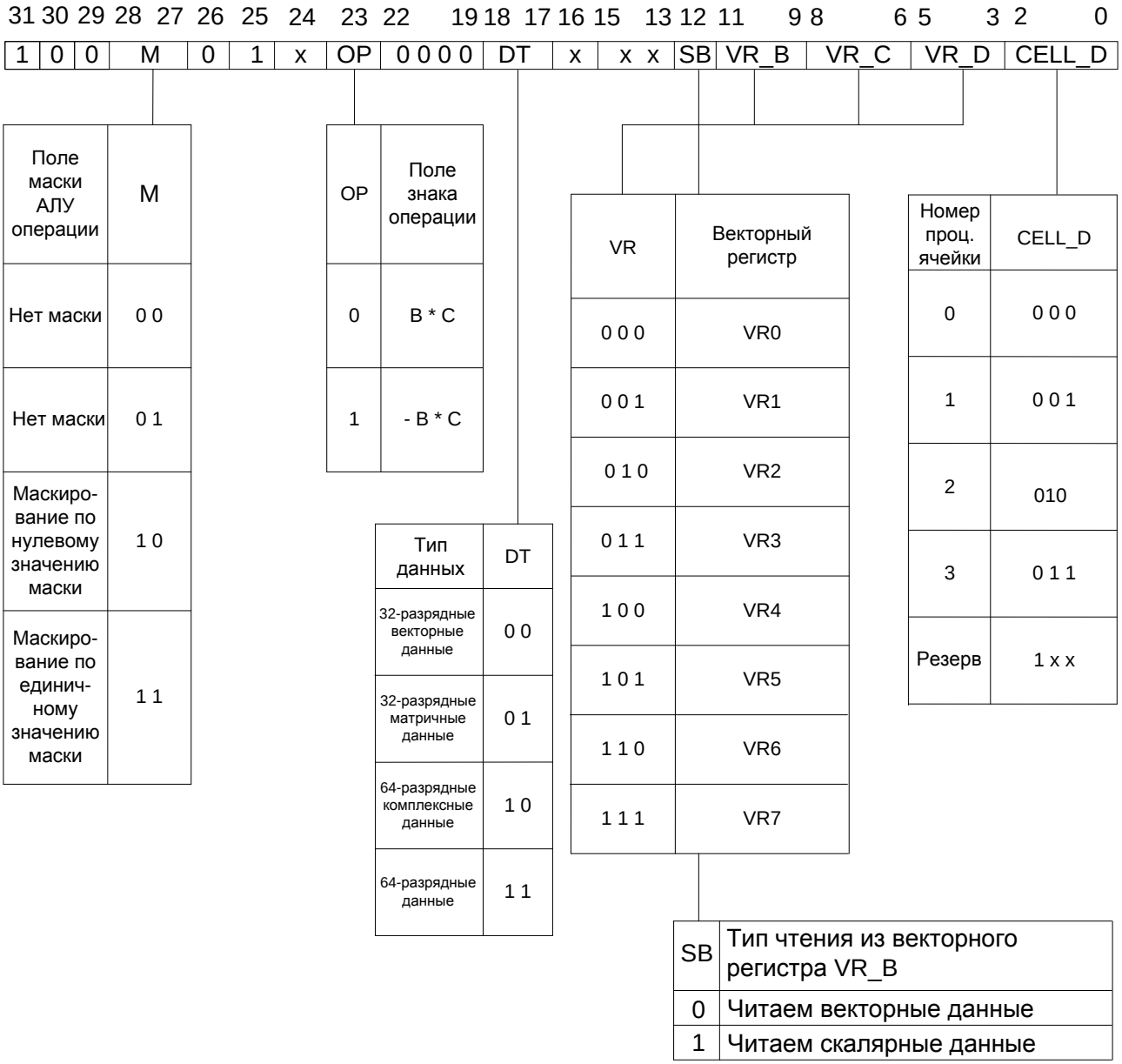
OP	Поле знака операции
0 0	A
0 1	|A|
1 0	- A
1 1	- |A|

Тип данных	DT
32-разрядные векторные данные	0 0
32-разрядные матричные данные	0 1
64-разрядные комплексные данные	1 0
64-разрядные данные	1 1

VR	Векторный регистр
0 0 0	VR0
0 0 1	VR1
0 1 0	VR2
0 1 1	VR3
1 0 0	VR4
1 0 1	VR5
1 1 0	VR6
1 1 1	VR7

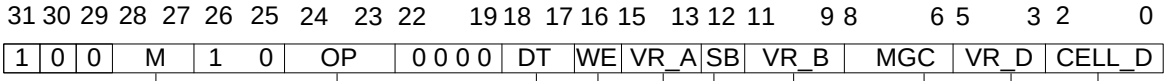
Номер проц. ячейки	CELL_D
0	0 0 0
1	0 0 1
2	010
3	0 1 1
Резерв	1 x x

Формат 3.2F (fap.(VR_B * VR_C)) → VR_D



Примечание - При использовании 32-разрядных матричных данных в качестве операнда B 11-10 разряды команды задают чтение регистровой пары (128 разрядов за такт, а не 64, как обычно). Это позволяет за один процессорный такт заполнить матрицу 2 x 2 32-разрядных элементов. 9 разряд команды задаёт, каким образом формируется матрица: если 9 разряд равен нулю, матрица считывается по столбцам (один столбец из векторного регистра с чётным номером, второй – из векторного регистра с нечётным номером), если 9 разряд равен единице – по строкам (одна строка берётся из векторного регистра с чётным номером, вторая – из векторного регистра с нечётным номером). В первом случае это позволяет осуществить умножение матрицы B на вектор-строку C слева, во втором - умножение матрицы B на вектор-столбец C справа (см. рисунок 3.9 и пояснения к нему).

Формат 3.3F (fap.(VR_A * VR_ B)) → VR_D



Поле маски АЛУ операции	M
Нет маски	0 0
Нет маски	0 1
Маскирование по нулевому значению маски	1 0
Маскирование по единичному значению маски	1 1

OP	Поле знака операции
0 0	A + B
0 1	A - B
1 0	- A + B
1 1	- A - B

VR	Векторный регистр
0 0 0	VR0
0 0 1	VR1
0 1 0	VR2
0 1 1	VR3
1 0 0	VR4
1 0 1	VR5
1 1 0	VR6
1 1 1	VR7

Номер проц. ячейки	CELL_D
0	0 0 0
1	0 0 1
2	010
3	0 1 1
Резерв	1 x x

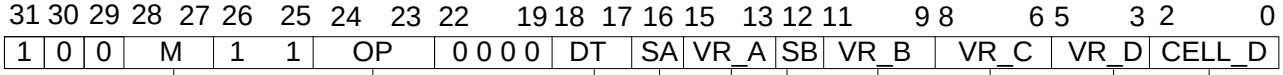
Тип данных	DT
32-разрядные векторные данные	0 0
32-разрядные матричные данные	0 1
64-разрядные комплексные данные	1 0
64-разрядные данные	1 1

Разрешение записи в VR_D в командах сравнения	WE
Нет записи	0
Есть запись	1

SB	Тип чтения из векторного регистра VR_B
0	Читаем векторные данные
1	Читаем скалярные данные

MGC	Условия генерации маски
0 0 0	Маска и признаки не формируются
0 0 1	Формируются только признаки
0 1 0	Формируются признаки и маска, если больше
0 1 1	Формируются признаки и маска, если меньше
1 0 0	Формируются признаки и маска, если равно
1 0 1	Формируются признаки и маска, если не равно
1 1 0	Формируются признаки и маска, если больше или равно
1 1 1	Формируются признаки и маска, если меньше или равно

Формат 3.4F (fap.(VR_A + VR_ B * VR_C)) → VR_D



31	30	29	28 27	26	25	24 23	22 19	18 17	16	15 13	12 11	9 8	6 5	3 2	0
1	0	0	M	1	1	OP	0 0 0 0	DT	SA	VR_A	SB	VR_B	VR_C	VR_D	CELL_D

Поле маски АЛУ операции	M
Нет маски	0 0
Нет маски	0 1
Маскирование по нулевому значению маски	1 0
Маскирование по единичному значению маски	1 1

OP	Поле знака операции
0 0	A + B
0 1	A - B
1 0	- A + B
1 1	- A - B

VR	Векторный регистр
0 0 0	VR0
0 0 1	VR1
0 1 0	VR2
0 1 1	VR3
1 0 0	VR4
1 0 1	VR5
1 1 0	VR6
1 1 1	VR7

Номер проц. ячейки	CELL_D
0	0 0 0
1	0 0 1
2	010
3	0 1 1
Резерв	1 x x

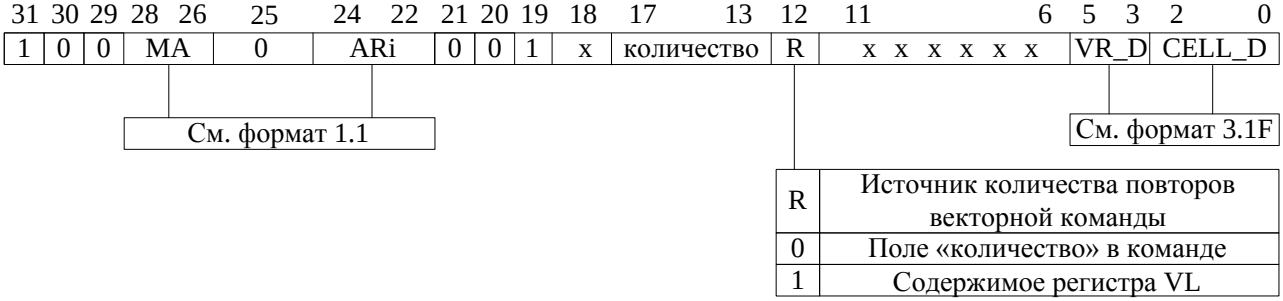
Тип данных	DT
32-разрядные векторные данные	0 0
32-разрядные матричные данные	0 1
64-разрядные комплексные данные	1 0
64-разрядные данные	1 1

S	Тип чтения из векторного регистра VR_A или VR_B
0	Читаем векторные данные
1	Читаем скалярные данные

Примечание - При использовании 32-разрядных матричных данных в качестве операнда В 11-10 разряды команды задают чтение регистровой пары (128 разрядов за такт, а не 64, как обычно). Это позволяет за один процессорный такт заполнить матрицу 2 x 2 32-разрядных элементов. 9 разряд команды задаёт, каким образом формируется матрица: если 9 разряд равен нулю, матрица считывается по столбцам (один столбец из векторного регистра с чётным номером, второй – из векторного регистра с нечётным номером), если 9 разряд равен единице – по строкам (одна строка берётся из векторного регистра с чётным номером, вторая – из векторного регистра с нечётным номером). В первом случае это позволяет осуществить умножение матрицы В на вектор-строку С слева, во втором - умножение матрицы В на вектор-столбец С справа (см. рисунок 3.9 и пояснения к нему).

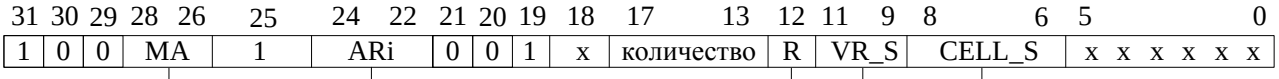
6.19 Векторные команды ввода-вывода сопроцессора арифметики с плавающей точкой

Формат 4.1F Чтение из памяти векторных данных VR_D <- (fадр.(ARi, GRi)); ARi←fм(ARi, GRi)



R	Источник количества повторов векторной команды
0	Поле «количество» в команде
1	Содержимое регистра VL

Формат 4.2F Запись в память векторных данных VR_D -> (fадр.(ARi, GRi)); ARi←fм(ARi, GRi)

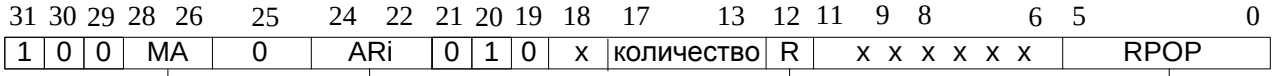


Источник количества повторов векторной команды	R
Поле «количество» в команде	0
Содержимое регистра VL	1

CELL_S	Номер проц. Ячейки-источника
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 x x	Резерв

Векторный регистр - источник	VR_S
VR0	0 0 0
VR1	0 0 1
VR2	0 1 0
VR3	0 1 1
VR4	1 0 0
VR5	1 0 1
VR6	1 1 0
VR7	1 1 1

Формат 4.3F Чтение из памяти векторных данных с последующим их преобразованием в блоке упаковки и распаковки данных: REPACK UNIT <- (fадр.(ARi, GRi)); ARi←fм(ARi, GRi)

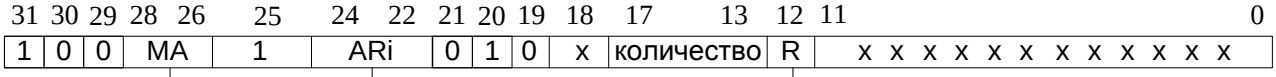


См. формат 1.1

Источник количества повторов векторной команды	R
Поле «количество» в команде	0
Содержимое регистра VL	1

Операции упаковки/распаковки	RPOP	
Форматы входных (SF) и выходных (преобразованных) данных (DF)	SF	DF
Два 32-разрядных числа в формате с фиксированной точкой	0 0 0	
Два 32-разрядных числа в формате с фиксированной точкой, но в дальнейшем используются только старшие 32-разряда (младшие 32-разряда обнуляются)	0 0 1	
Два 32-разрядных числа в формате с фиксированной точкой, но в дальнейшем используются только младшие 32-разряда (старшие 32-разряда обнуляются)	0 1 0	
Одно 64-разрядное число в формате с фиксированной точкой	0 1 1	
Два 32-разрядных числа в формате с плавающей точкой	1 0 0	
Два 32-разрядных числа в формате с плавающей точкой, но в дальнейшем используются только старшие 32-разряда (младшие 32-разряда обнуляются)	1 0 1	
Два 32-разрядных числа в формате с плавающей точкой, но в дальнейшем используются только младшие 32-разряда (старшие 32-разряда обнуляются)	1 1 0	
Одно 64-разрядное число в формате с плавающей точкой двойной точности	1 1 1	

Формат 4.4F Запись в память векторных данных из блока упаковки и распаковки данных: REPACK UNIT -> (fадр.(ARi, GRi)); ARi←fм(ARi, GRi)



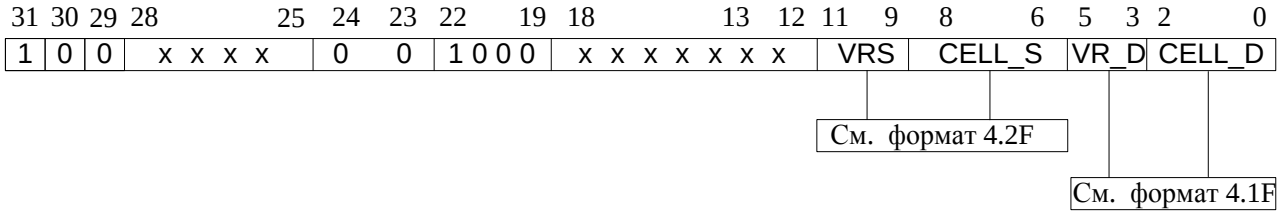
См. формат 1.1

Источник количества повторов векторной команды	R
Поле «количество» в команде	0
Содержимое регистра VL	1

Примечание: поле “Количество” задает количество 64-разрядных слов, участвующих в операциях пересылок данных:
00000 - одно 64-разрядное слово;
00001 - два 64-разрядных слова;
…
11111 – тридцать два 64-разрядных слова.

6.20 Векторные команды пересылки данных для сопроцессора арифметики с плавающей точкой

Формат 5.1F Пересылка данных из векторного регистра одной из процессорных ячеек в векторный регистр другой ячейки: VR_D (SEL_D) <- VR_S (SEL_S)



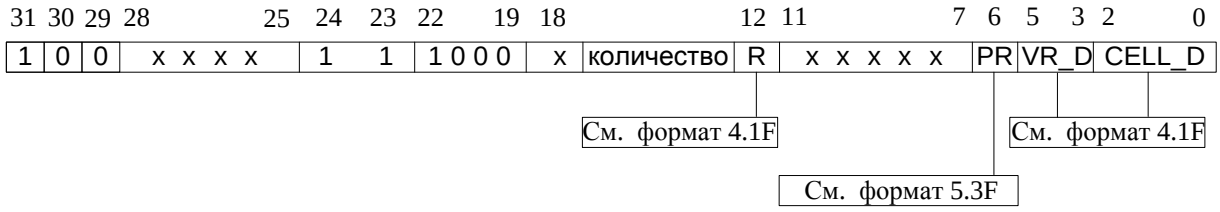
Формат 5.2F Пересылка данных в зависимости от маски из векторного регистра VRS0 или VRS1 одной из процессорных ячеек в векторный регистр другой ячейки: VR_D (SEL_D) <- fмаск. (VRS0 (SEL_S), VRS1 (SEL_S))

Векторный регистр - источник	VR_S
VR0	0 0 0
VR1	0 0 1
VR2	0 1 0
VR3	0 1 1
VR4	1 0 0
VR5	1 0 1
VR6	1 1 0
VR7	1 1 1

Формат 5.3F Пересылка данных из векторного регистра (пары векторных регистров) одной из процессорных ячеек с последующим их преобразованием в блоке упаковки и распаковки данных: REPACK UNIT <- VRS$_i$ (SEL_S), {VRS $_{i+1}$ (SEL_S)}

Регистровая пара	PR
Отдельный регистр (Регистровая пара не используется)	0
Регистровая пара используется	1

Формат 5.4F Пересылка векторных данных из блока упаковки и распаковки в векторный регистр (в пару векторных регистров) одной из процессорных ячеек: REPACK UNIT - >VRS$_i$ (SEL_S), {VRS $_{i+1}$ (SEL_S)}



6.21 Поле R$_{ист/пр-к}$ в командах пересылки данных сопроцессора арифметики с плавающей точкой

Таблица 6.3 - Поле Rист/пр-к в командах пересылки данных сопроцессора арифметики с плавающей точкой

Код регистра	Регистр- источник	Регистр- приемник
0 0 0 0 0 0	FPFR0	FPFR0
0 0 0 0 0 1	SPMRL0	SPMRL0
0 0 0 0 1 0	SPMRH0	SPMRH0
0 0 0 0 1 1	DPMR0	DPMR0
0 0 0 1 0 0	FPFR1	FPFR1
0 0 0 1 0 1	SPMRL1	SPMRL1
0 0 0 1 1 0	SPMRH1	SPMRH1
0 0 0 1 1 1	DPMR1	DPMR1
0 0 1 0 0 0	FPFR2	FPFR2
0 0 1 0 0 1	SPMRL2	SPMRL2
0 0 1 0 1 0	SPMRH2	SPMRH2
0 0 1 0 1 1	DPMR2	DPMR2
0 0 1 1 0 0	FPFR3	FPFR3
0 0 1 1 0 1	SPMRL3	SPMRL3
0 0 1 1 1 0	SPMRH3	SPMRH3
0 0 1 1 1 1	DPMR3	DPMR3
0 1 0 0 0 0	FPIOIR	FPIOIR
0 1 0 0 0 1	FPOIR	FPOIR
0 1 0 0 1 0	FPUIR	FPUIR
0 1 0 0 1 1	FPIIR	FPIIR
0 1 0 1 0 0	FPDLIR	FPDLIR
0 1 0 1 0 1	FPIEIR	FPIEIR
0 1 0 1 1 0	RIER	RIER
0 1 0 1 1 1	IRRreset	-
0 1 1 0 0 0	-	-
0 1 1 0 0 1	SIR	SIR
0 1 1 0 1 0	RND_00	FPCR
0 1 1 0 1 1	RND_01	FPCR
0 1 1 1 0 0	RND_10	FPCR
0 1 1 1 0 1	RND_11	FPCR
0 1 1 1 1 0	Preset	FPCR
0 1 1 1 1 1	Pset	FPCR

Таблица 6.4 - Обозначение регистров сопроцессора арифметики с плавающей точкой и их назначение

Обозначение регистров и их назначение	Разрядность
Регистры процессорных ячеек сопроцессора арифметики с плавающей точкой	
FPFRi– регистр флагов процессорной ячейки i (i=0,…,3)	4
SPMRLi – младшая часть регистра маски одинарной точности процессорной ячейки i (i=0,…,3)	32
SPMRHi – старшая часть регистра маски одинарной точности процессорной ячейки i (i=0,…,3)	32
DPMRi – псевдо-регистр маски двойной точности процессорной ячейки i (i=0,…,3). Запись в него 32-разрядного значения вызывает запись одного и того же в регистры SPMRLi и SPMRHi	32
Регистры контроллера прерываний сопроцессора арифметики с плавающей точкой	
FPIOIR - информация о прерывании по некорректным данным	6
FPOIR - информация о прерывании по переполнению	6
FPUIR - информация о прерывании по потере значимости	6
FPIIR - информация о прерывании по потере точности	6
FPDLIR - информация о прерывании по потере данных при сложении трёх операндов	6
FPIEIR - информация о прерывании по неправильной команде	10
RIER – регистр порядка целого числа при переупаковке	11
IRRreset –сброс всех запросов на прерывание сопроцессора арифметики с плавающей точкой	-
Общие регистры сопроцессора арифметики с плавающей точкой	
SIR – регистр системного интегратора, через который осуществляются пересылки между регистрами RISC-процессора и сопроцессоров	32
FPCR – регистр управления, содержащий бит параллельной работы и режимы округления	3
RND_00 – установка режима округления к ближайшему (по умолчанию) в регистре FPCR	-
RND_01 – установка режима округления к -∞ в регистре FPCR	-
RND_10 – установка режима округления к +∞ в регистре FPCR	-
RND_11 – установка режима округления к нулю в регистре FPCR	-
Preset – сброс бита параллельной работы в регистре FPCR	-
Pset – установ бита параллельной работы в регистре FPCR	-

7 Периферийные блоки процессорных систем NMPU0 и NMPU1

7.1 Общие сведения

В состав периферийных блоков каждой из процессорных систем NMPU0 и NMPU1 входят следующие узлы:

- Мост "системный интегратор - периферийная шина";
- Системный контроллер;
- Блок таймеров;
- Мост "системный интегратор - AXI";
- Блок защиты памяти;
- Блок управления кэш-памятью команд;
- Коммуникационные порты процессорной системы;
- Контроллер прерываний процессорной системы.

Периферийные блоки каждой процессорной системы управляются только своим процессорным ядром и недоступны другому ядру. Набор периферийных блоков, их назначение и функционирование, за исключением контроллера прерываний, одинаковы у обоих процессорных систем.

7.2 Мост "системный интегратор - периферийная шина"

Мост "системный интегратор - периферийная шина" обеспечивает процессорному ядру NMC4 доступ к собственным периферийным устройствам своей процессорной системы.

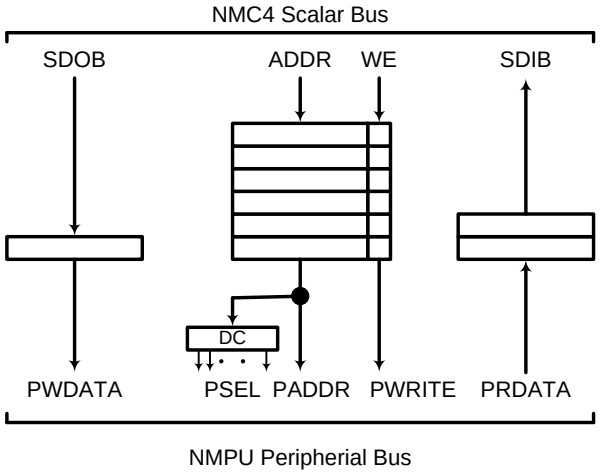


Рисунок 7.1 - Мост "системный интегратор - периферийная шина"

Мост состоит из буфера адресов, буфера даных чтения, регистра данных записи и схемы управления. Транзакция на периферийной шине занимает всегда 2 такта и не конвейризована.

Мост не имеет программно доступных регистров.

Доступ на периферийную шину допускается только скалярными командами загрузки и выгрузки регистров процессорного ядра NMC. При обращении по адресам периферийной шины векторными командами NMC поведение системы не специфицировано.

Все регистры на периферийной шине NMCell имеют чётные адреса, а все нечётные адреса зарезервированы. Это означает, что:

- Чтение возвращает значение 0;

- При записи регистровыми парами значение старшего 32-разрядного слова (РОН) отбрасывается, значение младшего (адресный регистр) записывается;
- При чтении регистровыми парами в старшем 32-разрядном слове (РОН) возвращается 0, в младшем (адресный регистр) - возвращается прочитанное значение.

7.3 Системный контроллер

Системный контроллер выполняет несколько функций:

- Содержит идентификатор процессорной системы;
- Служит для генерации запросов на прерывание к другим процессорным системам;
- Управляет выводами общего назначения процессорной системы;
- Настраивает скорость работы передающих коммуникационных портов.

Структурная схема системного контроллера представлена на Рисунок 7.2. Список программно доступных регистров системного контроллера представлен в Таблица 7.1.

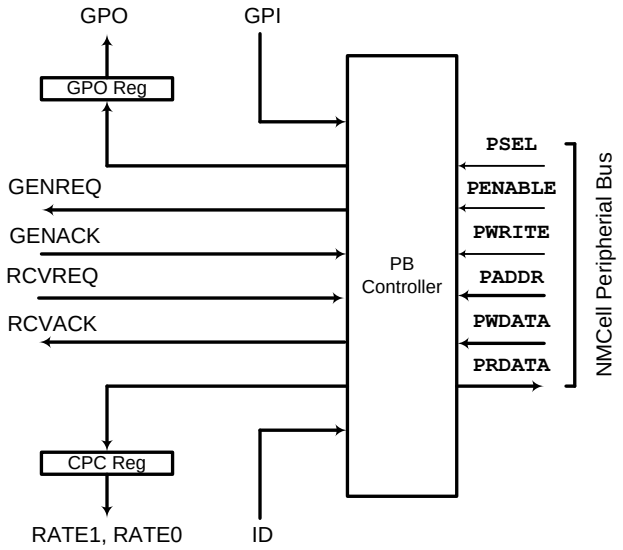


Рисунок 7.2 - Структурная схема системного контроллера

Таблица 7.1 - Программно доступные регистры системного контроллера

Адрес	Регистр	Доступ	Описание
4000_0000h	ID	ЧТ	Регистр идентификации процессорной системы.
4000_0002h	SYNC	ЧТ/ЗП	Регистр межпроцессорной синхронизации.
4000_0004h	GPIO	ЧТ/ЗП	Регистр входов и выходов общего назначения процессорной системы NMCell. Запись устанавливает значение на выходах общего назначения, чтение возвращает текущее состояние входов общего назначения.
4000_0006h	CPC	ЧТ/ЗП	Регистр управления передающей частью коммуникационного порта.

7.3.1 Регистр идентификации процессорной системы (ID)

Регистр идентификации процессорной системы доступен только для чтения. Регистр содержит 32-разрядный идентификатор процессорной системы.

Таблица 7.2 - Формат регистра идентификации процессорной системы ID

Разряды	Название	Доступ	Описание
[31:24]	INDEX	ЧТ	Номер процессорной системы в составе СБИС: 00000000 – NMPU0; 00000001 – NMPU1.
[23:20]	VAR	ЧТ	Особенности архитектуры процессорного ядра: 0010 - ядро с сопроцессором с плавающей точкой, 0001 - ядро с сопроцессором с фиксированной точкой.
[19:16]	ARCH	ЧТ	Архитектура ядра: 0100 - NMC4.
[15:4]	PRTNUM	ЧТ	Идентификатор изделия: 0x0B7 - микросхема.
[3:0]	REV	ЧТ	Версия изделия: 0000.

7.3.2 Регистр межпроцессорной синхронизации (SYNC)

Регистр межпроцессорной синхронизации позволяет процессорным системам обмениваться сигналами запроса и подтверждения. Поле GENREQ отвечает за выдачу запросов другим процессорным системам, поле RCVREQ используется для приёма запросов и выдачи подтверждений.

Таблица 7.3 - Формат регистра межпроцессорной синхронизации SYNC

Разряды	Название	Доступ	Описание
31-24	-	-	Зарезервировано
23-16	GENREQ	ЧТ/ЗП	Один бит поля GENREQ соответствует одному биту поля RCVREQ другой процессорной системы. Запись значения 1 в бит поля GENREQ вызывает аппаратную установку соответствующего бита RCVREQ другой процессорной системы. Программный сброс бита RCVREQ другой процессорной системы сбрасывает аппаратно бит поля GENREQ.
15-8	-	-	Зарезервировано
7-0	RCVREQ	ЧТ/ЗП	Один бит поля RCVREQ соответствует одному биту GENREQ другой процессорной системы. Программная установка соответствующего бита GENREQ в другой процессорной системе устанавливает бит GENREQ. Запись значения 1 в бит поля RCVREQ сбрасывает этот бит и инициирует аппаратный сброс соотвествующего бита GENREQ. Установка бита поля RCVREQ вызывает запрос на прерывание.

Таблица 7.4 - Соответствие разрядов полей регистра SYNC и запросов на прерывание

Разряд регистра SYNC запросчика	Разряд регистра SYNC приёмника	Запрос на прерывание в приёмнике
23 (GENREQ[7])	7 (RCVREQ[7])	Прерывание 7
22 (GENREQ[6])	6 (RCVREQ[6])	Прерывание 6
21 (GENREQ[5])	5 (RCVREQ[5])	Прерывание 5
20 (GENREQ[4])	4 (RCVREQ[4])	Прерывание 4
19 (GENREQ[3])	3 (RCVREQ[3])	Прерывание 3
18 (GENREQ[2])	2 (RCVREQ[2])	Прерывание 2
17 (GENREQ[1])	1 (RCVREQ[1])	Прерывание 1
16 (GENREQ[0])	0 (RCVREQ[0])	Прерывание 0

7.3.3 Регистр входов и выходов общего назначения (GPIO)

Регистр входов и выходов общего назначения управляет состояниями выходов общего назначения своей процессорной системы и позволяет считывать состояния входов общего назначения этой системы. Имеется 8 выходов и 8 входов общего назначения. Регистр для входов и выходов общий: запись в регистр устанавливает значение на выходах общего назначения, чтение возвращает текущее состояние входов общего назначения. Соответственно, состояние выходов не доступно для чтения. Значение всех выходов при сбросе - 0.

Таблица 7.5 - Формат регистра входов и выходов общего назначения GPIO

Разряды	Доступ	Описание
31-8	-	Зарезервировано
7	ЧТ/ЗП	Запись устанавливает значение выхода 7 общего назначения, чтение возвращает значение на входе 7 общего назначения.
6	ЧТ/ЗП	Запись устанавливает значение выхода 6 общего назначения, чтение возвращает значение на входе 6 общего назначения.
5	ЧТ/ЗП	Запись устанавливает значение выхода 5 общего назначения, чтение возвращает значение на входе 5 общего назначения.
4	ЧТ/ЗП	Запись устанавливает значение выхода 4 общего назначения, чтение возвращает значение на входе 4 общего назначения.
3	ЧТ/ЗП	Запись устанавливает значение выхода 3 общего назначения, чтение возвращает значение на входе 3 общего назначения.
2	ЧТ/ЗП	Запись устанавливает значение выхода 2 общего назначения, чтение возвращает значение на входе 2 общего назначения.
1	ЧТ/ЗП	Запись устанавливает значение выхода 1 общего назначения, чтение возвращает значение на входе 1 общего назначения.
0	ЧТ/ЗП	Запись устанавливает значение выхода 0 общего назначения, чтение возвращает значение на входе 0 общего назначения.

7.3.4 Регистр управления передающей частью коммуникационного порта (CPC)

Регистр управления передающей частью коммуникационного порта управляет скоростью передачи на внешних интерфейсах коммуникационных портов процессорной системы.

Таблица 7.6 - Формат регистра управления передающей частью коммуникационного порта СРС

Разряды	Название	Доступ	Описание
31-8	-		Зарезервировано
9	CDC1	ЧТ/ЗП	Режим управления выходными буферами коммуникационного порта 1.
8	CDC0	ЧТ/ЗП	Режим управления выходными буферами коммуникационного порта 0.
7-4	RATE1	ЧТ/ЗП	Скорость передачи коммуникационного порта 1.
3-0	RATE0	ЧТ/ЗП	Скорость передачи коммуникационного порта 0.

Поле RATE0 (разряды 3-0) задаёт скорость передачи коммуникационного порта 0, поле RATE1 (разряды 7-4) - коммуникационного порта 1. Допустимые значения полей RATE0 и RATE1: 0011b – 125 Мбайт/с, 0100b – 100 Мбайт/с, 0101b – 83,33 Мбайт/с (при тактовой частоте работы процессора – 500 МГц). Не допускается запись в данное поле других значений. Принимающие каналы коммуникационных портов не имеют настройки скорости, так как приём ведётся на скорости передатчика.

Биты CDC1 и CDC0 задают режим управления выходными буферами коммуникационных портов. Значение 0 – унаследованный режим (с перекрытием драйверов линии STRB), значение 1 – режим с высокоимпедансным состоянием линии STRB.

7.4 Блок таймеров

В состав периферийных узлов процессорной системы входят два независимых 32-разрядных таймера, которые пользователь может использовать как для отсчета задаваемых интервалов времени, так и для счета событий, являющихся внешними по отношению к процессорной системе. Структура блока таймеров представлена на Рисунок 7.3.

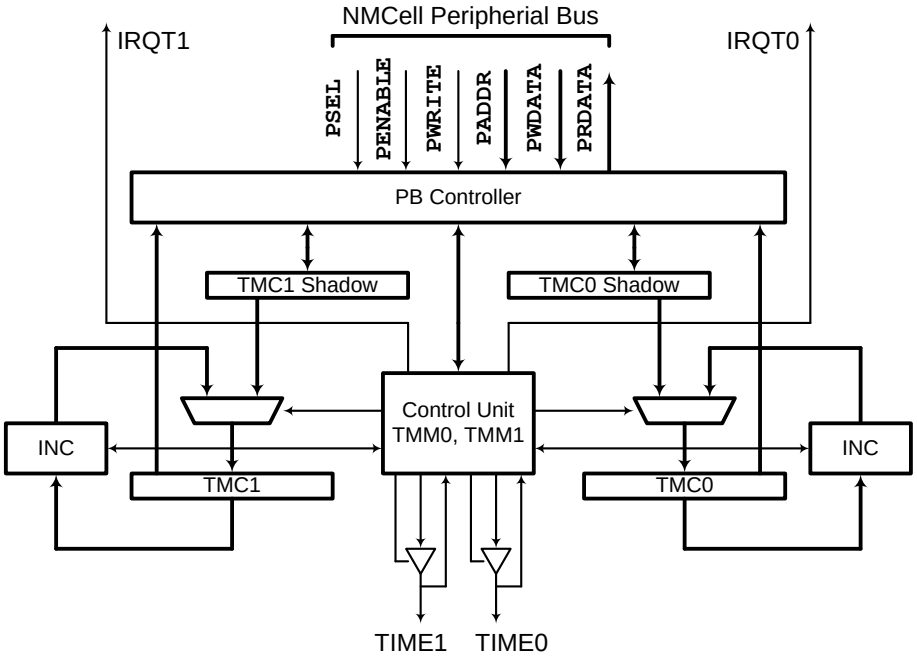


Рисунок 7.3 - Структурная схема блока таймеров

Таймер может работать в одном из двух режимов: в режиме однократного запуска и в непрерывном режиме. Интервал счета таймера задается программно. В качестве сигнала счета можно выбрать тактовый сигнал процессора или внешний сигнал, подаваемый на соответствующий вывод микросхемы.

По достижении нулевого значения таймер формирует запрос на прерывание, который может быть обработан стандартным образом. Кроме того, при достижении нулевого значения таймер может аппаратно формировать управляющий сигнал на внешних выводах микросхемы.

С блоком таймеров связаны следующие внешние выводы микросхемы: UiTIME0 и UiTIMER1 (i = 0 для NMPU0, 1 для NMPU1). При этом вывод UiTIME0 жестко связан с таймером 0, а вывод UiTIME1 с таймером 1. Каждый из выводов двунаправлен и может использоваться как вход и как выход, в зависимости от режима работы соответствующего таймера.

Каждый таймер в блоке таймеров имеет 2 регистра: регистр счётчика и регистр настройки.

Таблица 7.7 - Программно доступные регистры блока таймеров

Адрес	Название	Доступ	Описание
4000_0800h	TMC0	ЧТ/ЗП	Регистр счётчика таймера 0
4000_0802h	TMM0	ЧТ/ЗП	Регистр настройки таймера 0
4000_0804h	TMC1	ЧТ/ЗП	Регистр счётчика таймера 1
4000_0806h	TMM1	ЧТ/ЗП	Регистр настройки таймера 1

Запись в регистр счётчика производится через теневой регистр. Теневой регистр доступен для записи всегда. Копирование значения из теневого регистра в рабочий происходит по любому из условий:

- При запуске таймера (запись значения 1 в бит TA регистра настройки);
- В момент переключения, если текущее значение таймера равно нулю.

При чтении всегда возвращается значение рабочего регистра.

7.4.1 Регистр настройки таймера (TMMx)

Каждый таймер имеет свой регистр настройки. Регистр доступен на запись и чтение.

Таблица 7.8 - Формат регистра настройки таймера

Разряды	Название	Доступ	Описание
[3:2]	TP[1)]	ЧТ/ЗП	Управление внешним выводом TIMERx: 00 – вывод TIMERx работает как вход счетного сигнала таймера, 01 - вывод TIMERx работает как выход и меняет свое состояние при каждом обнулении таймера, 10 - вывод TIMERx работает как выход и, когда счётчик проходит значение 0, на вывод подаётся импульс положительной полярности длительностью N*** такта процессорной системы NMCell, 11 - вывод TIMERx работает как выход, когда счётчик проходит значение 0, на вывод подаётся импульс отрицательной полярности длительностью N*** такта.
[1]	TM	ЧТ/ЗП	Режим работы таймера: 0 – однократный режим работы таймера, 1 - непрерывный режим работы таймера.
[0]	TA	ЧТ/ЗП	Разрешение работы таймера: 0 – таймер остановлен, 1 - таймер осуществляет счет.

1) После системного сброса внешний вывод таймера сконфигурирован как вход

7.4.2 Режимы работы таймеров

Таймер может работать в одном из двух режимов:

- В режиме отсчёта временных интервалов;
- В режиме счёта внешних событий.

На Рисунок 7.4 представлены временные диаграммы работы таймера в режиме отсчета временных интервалов.

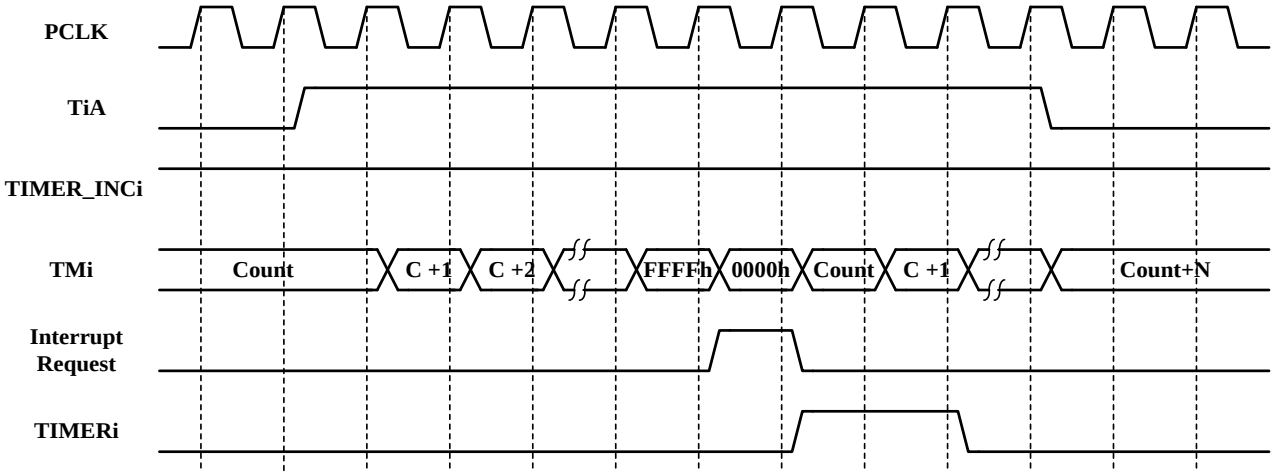


Рисунок 7.4 - Временные диаграммы работы таймера в режиме отсчета временных интервалов

После разрешения работы таймера, по следующему положительному фронту тактового сигнала процессорной системы, начальное значение, соответствующее отсчитываемому временному интервалу, переписывается из теневого регистра таймера в рабочий. После этого рабочий регистр таймера инкрементируется в каждом процессорном такте. При обнулении рабочего счетчика формируется сигнал Interrupt Request, который обрабатывается блоком прерываний стандартным образом. Одновременно с этим, если задан непрерывный режим работы таймера, из теневого регистра в рабочий переписывается значение, определяющее отсчитываемый временной интервал. Если задан однократный режим работы таймера, то счет прекращается, и соответствующий бит TA регистра настройки сбрасывается.

В следующем процессорном такте после достижения нулевого значения таймера изменяется сигнал на выводе TIMERx процессора.

Длительность временного интервала, отсчитываемого таймером, задается в дополнительном коде. Таким образом, длительность отсчитываемого временного интервала может изменяться от одного (код FFFFh) до 2^{32} (код 0000h) процессорных такта.

На Рисунок 7.5 представлены временные диаграммы работы таймера в режиме счета внешних событий.

При разрешении работы таймера значение теневого регистра таймера переписывается в рабочий так же, как и при работе в режиме отсчета временных интервалов.

Положительный фронт сигнала на входе TIMERx фиксируется процессором. Через два процессорных такта инкрементируется рабочий регистр таймера. В остальном работа таймера в данном режиме аналогична работе таймера в режиме отсчета временных интервалов.

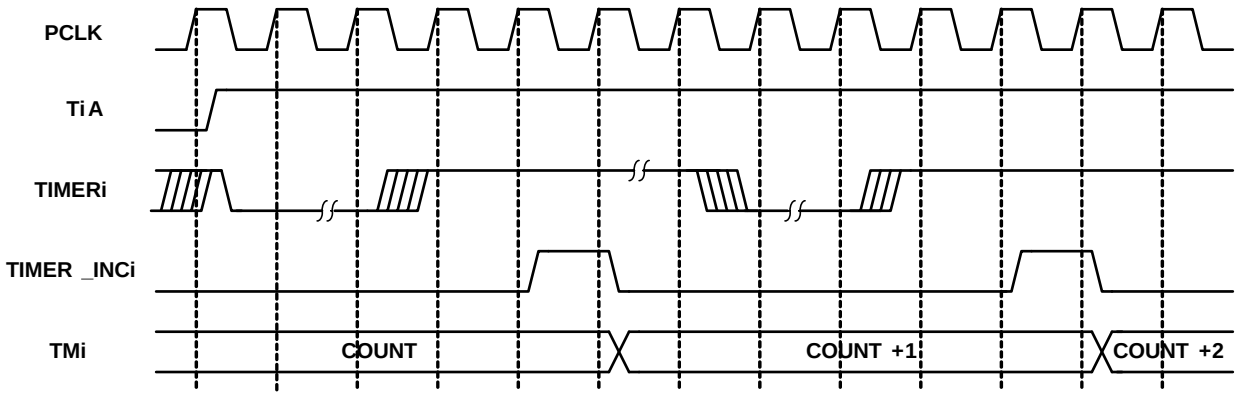


Рисунок 7.5 - Временные диаграммы работы таймера в режиме счета внешних событий

7.5 Мост "системный интегратор - AXI"

7.5.1 Состав и структура моста

Мост "системный интегратор - AXI" обеспечивает процессорному ядру NMC4 доступ к внешней памяти и периферийным устройствам системы на кристалле.

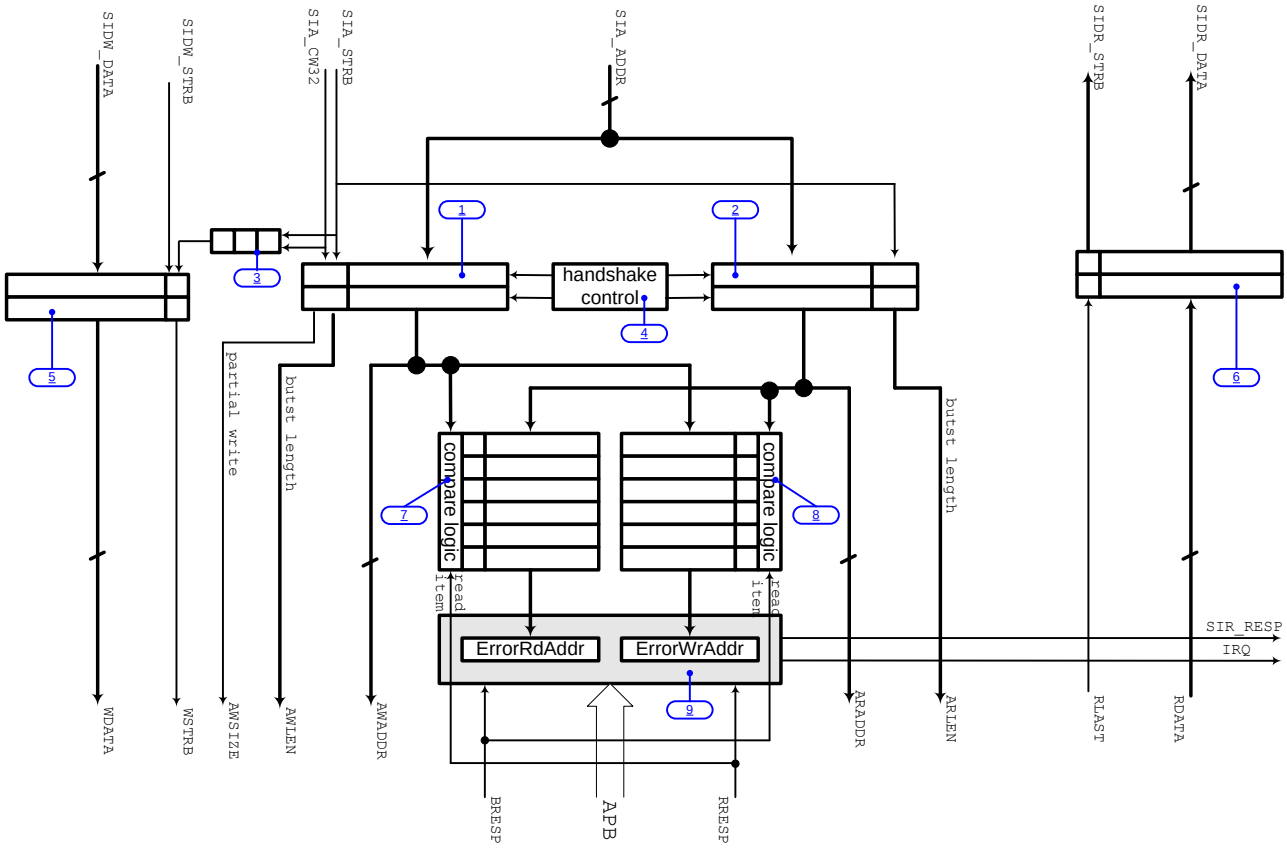


Рисунок 7.6 - Структурная схема моста "системный интегратор - AXI"

Элементы архитектуры моста:

1) буфер адресов записи,
2) буфер адресов чтения,
3) буфер типа операции записи - хранит признак: 64- или 32-разрядные данные, а также младший бит адреса,
4) блок управления – обрабатывает сигналы управления всей схемы, отслеживает порядок запросов,
5) буфер данных записи,
6) буфер данных чтения,
7) буфер сравнения для адресов чтения – сравнивает текущий адрес записи со всеми адресами чтения ещё не завершённых транзакций,
8) буфер сравнения для адресов записи - сравнивает текущий адрес чтения со всеми адресами записи ещё не завершённых транзакций,
9) блок обработки подтверждений – содержит программно доступные регистры.

Функции и особенности моста:

- Мост преобразует транзакции записи и чтения ядра NMC в транзакции записи и чтения шины AXI;
- Мост отслеживает порядок транзакций чтения и записи, и в случае конфликта по адресам двух транзакций в разных каналах (между чтением и записью) выдача более поздней транзакции блокируется до прихода подтверждения первой транзакции. Это обеспечивает исходный порядок (в котором получены адреса от NMC) операций доступа в память;
- Обращения в периферийную область (10000000h-3FFFFFFFh в 32-разрядной адресации) обрабатываются особым образом - исходный порядок обеспечивается для всех транзакций, попадающих в периферийную область;
- Мост различает запросы чтения данных и инструкций процессорного ядра. На шине AXI мост использует разные идентификаторы (ARID и RID): 0 – чтение данных, 1 – выборка инструкций. Адреса инструкций не проходят через буфер сравнения и с ними не сравнивается текущий адрес записи, то есть, если одновременно по одному адресу выполняется транзакция записи и транзакция выборки инструкции, то порядок их исполнения не гарантируется. При чтении данных и выборке инструкций поддерживается интерливинг данных чтения (спецификация AXI, п. 8.3 Read Ordering) неограниченной глубины: данные передаются к системному интегратору в том порядке, в котором получены с шины AXI (системный интегратор различает их по идентификатору, который передаётся вместе с данными);
- Мост корректно обрабатывает транзакции, которые возвращают ошибку в фазе подтверждения (сигналы RRESP, BRESP). Мост сохраняет адрес первой транзакции, закончившейся с ошибкой, и генерирует запрос на прерывание. Адрес ошибочной транзакции и функция управления прерыванием доступны для ядра NMC через периферийный интерфейс. Исключение составляет выборка инструкций – для неё не сохраняется адрес ошибочной транзакции;
- Мост использует только один тип транзакции AXI - одиночная транзакция типа INCR, при этом запись и чтение могут производиться 32- или 64-разрядными словами (все адреса, соответственно, выровнены до четырёх байт);
- Мост не использует идентификаторы AXI (в системе ему назначается постоянный идентификатор);
- Имеются 2 режима записи: режим запаздывающих данных и режим выровненных данных. В режиме выровненных данных адрес выдаётся на шину AXI только при наличии данных для записи. В режиме запаздывающих данных адрес выдаётся до появления данных, в таком случае время, на которое запаздывают данные, зависит от состояния конвейера ядра NMC и может достигать десятков тактов.

7.5.2 Особенности работы моста

Основная функция моста - распределение потока обращений в память на 2 независимых конвейера шины AXI: конвейер чтения и конвейер записи. Порядок выполнения операций в разных конвейерах не гарантируется, поэтому в случаях, когда порядок требуется соблюдать, мост обеспечивает порядок путём задержки более позднего запроса.

Порядок, в котором выполняются 2 операции, требуется соблюдать в следующих случаях:

- Запись в память, затем чтение по тому же адресу - чтение дожидается завершения записи, чтобы вернуть обновлённые данные;
- Чтение из памяти, затем запись по тому же адресу - запись дожидается завершения чтения, чтобы данные чтения не были обновлены;

- Запись в область периферийных устройств, затем чтение из области периферийных устройств;
- Чтение из области периферийных устройств, затем запись в область периферийных устройств - независимо от адреса при обращении к периферийным устройствам требуется соблюдать порядок операций чтения и записи.

Транзакция чтения AXI производится в 2 фазы: фаза адреса и фаза данных (в фазе данных приходит также сигнал подтверждения). В первой фазе адрес из буфера адресов выдаётся на шину AXI и одновременно записывается в буфер сравнения для адресов чтения. Мост поддерживает выдачу нескольких адресов чтения до получения данных, поэтому адреса чтения могут выдаваться до тех пор, пока не заполнится буфер сравнения. Адрес покидает буфер сравнения в момент прихода данных (фаза данных). Мост не поддерживает пакетные передачи, поэтому одному адресу всегда соответствует одно данное.

Транзакция записи AXI производится в 3 фазы: фаза адреса, фаза данных и фаза подтверждения. Мост имеет 2 режима записи: отличие их состоит в том, как соотносятся фаза адреса и фаза данных. В режиме выровненных данных адрес записи не выставляется на шину AXI и остаётся в буфере до прихода соответствующих данных записи. В режиме запаздывающих данных адрес выдаётся на шину AXI независимо от того, пришли на данный момент данные или нет.

Преимущество режима запаздывающих данных:

- Меньше блокировок ядра - например, при выполнении трёх операций записи, а затем чтения, если первая запись блокируется, то блокируется и чтение, так как не может обойти третью запись и попасть в буфер адресов чтения.

Преимущество режима выровненных данных:

- Меньше блокировок в конвейере записи в память - например, когда контроллер памяти принимает адреса записи от одной процессорной системы, он вынужден дожидаться соответствующих данных и не давать доступа другим устройствам системы: если данные приходят вместе с адресом.

Выдаваемый адрес записи поступает в буфер сравнения для адресов записи. Адрес покидает буфер сравнения в момент прихода подтверждения.

Каждая транзакция AXI завершается подтверждением, которое может быть (в данном случае) трёх типов:

- Нормальное завершение;
- Ошибка обращения к конечному устройству;
- Ошибка адресации.

Если в фазе подтверждения транзакции приходит сигнал ошибки, то код ошибки и адрес (читается из буфера сравнения) сохраняются в программно доступных регистрах моста. При возникновении ошибки мост может устанавливать сигнал запроса на прерывание.

7.5.3 Программно доступные регистры моста "системный интегратор - AXI"

Программно доступные регистры моста используются для обработки ошибок обращения в память и для настройки режима записи.

Таблица 7.9 - Программно доступные регистры моста «системный интегратор - AXI»

Адрес	Название	Доступ	Описание
4000_1000h	CSR	ЧТ/ЗП	Регистр управления
4000_1002h	IRQMask	ЧТ/ЗП	Маска прерывания
4000_1004h	RdAddr	ЧТ	Регистр адреса чтения
4000_1006h	WrAddr	ЧТ	Регистр адреса записи

7.5.3.1 Регистр адреса чтения и регистр адреса записи (RdAddr и WrAddr)

Регистр адреса (RdAddr и WrAddr) обновляется с завершением каждой транзакции, пока не установлен бит ошибки (ERD и EWR). Таким образом, при возникновении ошибки в нём остаётся адрес первой транзакции, завершившейся с ошибкой.

Таблица 7.10 - Формат регистров адреса чтения и записи (RdAddr и WrAddr)

Разряды	Название	Доступ	Описание
[31:30]	-	-	Зарезервировано
[29:0]	ADDR	ЧТ	Адрес завершившейся операции.

7.5.3.2 Регистр маски прерываний (IRQMask)

Регистр маски прерываний разрешает или запрещает выдачу запроса на прерывание к процессорному ядру NMC4.

Таблица 7.11 - Формат регистра маски прерываний (IRQMask)

Разряды	Название	Доступ	Описание
[2]	IRI	ЧТ/ЗП	Включение прерывания при ошибках выборки инструкций: 0 - прерывание запрещено, 1 - прерывание разрешено.
[1]	IWR	ЧТ/ЗП	Включение прерывания при ошибках записи: 0 - прерывание запрещено, 1 - прерывание разрешено.
[0]	IRD	ЧТ/ЗП	Включение прерывания при ошибках чтения данных: 0 - прерывание запрещено, 1 - прерывание разрешено.

Запрос на прерывание от моста установлен всегда, когда установлены биты ERD и IRD, ERI и IRI или EWR и IWR. Запрос снимается программно путём сброса битов ERD, ERI или EWR.

7.5.3.3 Регистр управления (CSR)

Регистр всегда доступен для чтения, и записи значения 1 в некоторые биты (ERD, EWR, ERI, MSET, MCLR).

Бит M имеет теневой регистр: запись значения 1 в биты MSET и MCLR изменяет только теневой регистр. Перезапись в рабочий регистр осуществляется автоматически, но только во время простоя канала записи (когда бит AWR имеет значение 0). Чтобы изменить значение бита M, необходимо провести запись значения 1 в бит MSET либо MCLR и дождаться, пока в бите M установится требуемое значение.

Признак ошибки (ERD, ERI и EWR) устанавливается аппаратно, если возвращаемый код подтверждения транзакции AXI (RRESP и BRESP, соответственно) равен 10b или 11b.

Поле кода подтверждения (RRD, RRI и RWR) обновляется с завершением каждой транзакции, пока не установлен соответствующий бит ошибки (ERD, ERI и EWR). Таким образом, в нём остаётся код первой произошедшей ошибки.

Таблица 7.12 - Формат регистра управления (CSR)

Разряды	Название	Доступ	Описание
[13]	AWR	ЧТ	Активность канала записи: 0 - мост не содержит команд записи и нет незавершённых транзакций записи, 1 - имеется хотя бы одна незавершённая команда записи. Данный бит не учитывает признак ошибки.
[12]	ARD	ЧТ	Активность канала чтения: 0 - мост не содержит команд чтения и нет незавершённых транзакций чтения, 1 - имеется хотя бы одна незавершённая команда чтения. Данный бит не учитывает признак ошибки.
[11:10]	RRI	ЧТ	Код подтверждения ошибочной операции при выборке команд (RRESP шины AXI): 00 - нормальное завершение (поле имеет значение 00, пока бит ERI не установлен), 01 - зарезервировано, 10 - конечное устройство вернуло ошибку, 11 - обращение в зарезервированную область памяти.
[9:8]	RWR	ЧТ	Код подтверждения ошибочной операции при записи данных (BRESP шины AXI). 00 - нормальное завершение (поле имеет значение 00, пока бит EWR не установлен), 01 - зарезервировано, 10 - конечное устройство вернуло ошибку, 11 - обращение в зарезервированную область памяти.
[7:6]	RRD	ЧТ	Код подтверждения ошибочной операции при чтении данных (RRESP шины AXI). 00 - нормальное завершение (поле имеет значение 00, пока бит ERD не установлен), 01 - зарезервировано, 10 - конечное устройство вернуло ошибку, 11 - обращение в зарезервированную область памяти.
[5]	MCLR	ЧТ/ЗП	Установка режима записи 0. Запись значения 1 в этот бит устанавливает режим записи 0.
[4]	MSET	ЧТ/ЗП	Установка режима записи 1. Запись значения 1 в этот бит устанавливает режим записи 1.
[3]	M	ЧТ	Текущий режим записи: 0 – режим запаздывающих данных, 1 – режим выровненных данных.
[2]	ERI	ЧТ/ЗП	Признак ошибки адресации в запросе на выборку команд. Устанавливается аппаратно в момент прихода подтверждения RRESP на шине AXI. Сброс производится программно путём записи значения 1 в данный бит.
[1]	EWR	ЧТ/ЗП	Признак ошибки адресации в запросе записи. Устанавливается аппаратно в момент прихода подтверждения BRESP на шине AXI. Сброс производится программно путём записи значения 1 в данный бит.
[0]	ERD	ЧТ/ЗП	Признак ошибки адресации в запросе чтения данных. Устанавливается аппаратно в момент прихода подтверждения RRESP на шине AXI. Сброс производится программно путём записи значения 1 в данный бит.

7.6 Блок защиты памяти

Блок защиты памяти разрешает доступ внешним устройствам в выделенные области памяти процессорной системы.

Внешние устройства имеют произвольный доступ к внутренней памяти процессорной системы через интерфейс AXI3 Slave. Блок защиты памяти находится внутри процессорной системы и непосредственно подключен к интерфейсу AXI3 Slave.

Блок защиты памяти декодирует адрес транзакции и определяет, разрешён или запрещён доступ по этому адресу. Блок пропускает разрешённые транзакции без изменения, а запрещённые транзакции обрабатывает следующим образом:

- С точки зрения внешнего устройства - транзакция завершается с кодом ошибки SLVERR, при этом в транзакции чтения вместо данных во всех передачах транзакции возвращаются нулевые значения;
- С точки зрения процессорной системы - генерируется (маскируемый по каждой выделенной области) запрос на прерывание к процессору NMC, который снимается программно.

Интерфейсы чтения и записи AXI3 разделены, поэтому в процессорной системе присутствуют 2 экземпляра блоков защиты памяти: по одному на чтение и на запись.

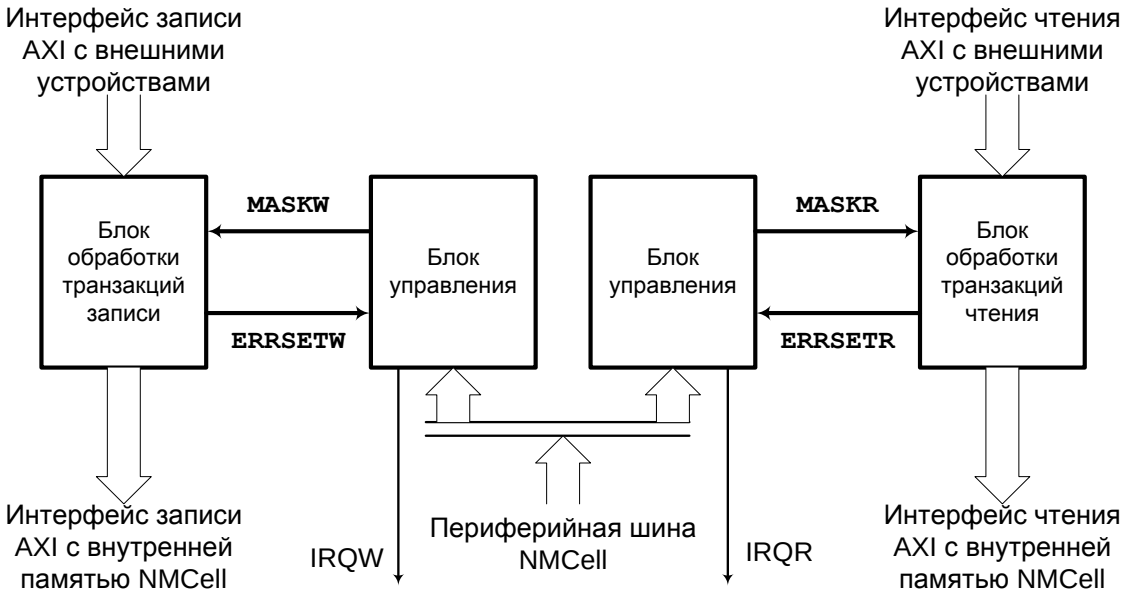


Рисунок 7.7 - Структурная схема блока защиты памяти

Каждый экземпляр блока защиты памяти состоит из двух частей:

- Блока управления - который содержит конфигурационные регистры устройства и генерирует сигнал (MASK[31:0]) маски доступных областей памяти;
- Блока обработки транзакций - который работает с сигналами шины AXI3, выдаёт сигнал (ERRSET[31:0]) обращения в запрещённую область.

Блок обработки транзакций производит сравнение адресов и самостоятельно завершает транзакции AXI3, адреса которых попадают в запрещённые сегменты.

В обычном режиме блок обработки транзакций не вмешивается в обработку проходящих транзакций, только ведёт подсчёт текущего количества незавершённых транзакций (для которых прошла передача адреса и ещё не было передачи последних данных чтения). Как только на входном интерфейсе блока появляется адрес, попадающий в защищённую область,

блок блокирует канал адреса, дожидается передачи всех данных от незавершённых транзакций и после этого выдаёт нулевые данные и код ошибки (RRESP=10b, ошибка конечного устройства).

Блок обработки транзакций сообщает блоку управления об обращении в защищённый сегмент с помощью шины ERRSETx. Блок управления фиксирует сигналы ERRSETx, в зависимости от настройки может генерировать запрос на прерывание к ядру NMC4.

7.6.1 Защищаемые сегменты внутренней памяти

Внутренняя память процессорной системы разделяется на 32 защищаемых сегмента, каждый размером 32 Кбайт. Сегменту с номером i соответствует один бит с номером i в каждом регистре управления блоком. В Таблица 7.13 представлена нумерация сегментов.

Таблица 7.13 - Защищаемые сегменты внутренней памяти

Сег-мент	Адреса в собственном пространстве NMCell (32-разрядный доступ)	Адреса в карте памяти внешних NMCell (32-разрядный доступ)	Адреса в карте памяти устройств ввода-вывода (байтовый доступ)
0	0000_0000h-0000_1FFFh	0004_0000h-0004_1FFFh	0010_0000h-0010_7FFFh
1	0000_2000h-0000_3FFFh	0004_2000h-0004_3FFFh	0010_8000h-0010_FFFFh
2	0000_4000h-0000_5FFFh	0004_4000h-0004_5FFFh	0011_0000h-0011_7FFFh
3	0000_6000h-0000_7FFFh	0004_6000h-0004_7FFFh	0011_8000h-0011_FFFFh
4	0000_8000h-0000_9FFFh	0004_8000h-0004_9FFFh	0012_0000h-0012_7FFFh
5	0000_A000h-0000_BFFFh	0004_A000h-0004_BFFFh	0012_8000h-0012_FFFFh
6	0000_C000h-0000_DFFFh	0004_C000h-0004_DFFFh	0013_0000h-0013_7FFFh
7	0000_E000h-0000_FFFFh	0004_E000h-0004_FFFFh	0013_8000h-0013_FFFFh
8	0001_0000h-0001_1FFFh	0005_0000h-0005_1FFFh	0014_0000h-0014_7FFFh
9	0001_2000h-0001_3FFFh	0005_2000h-0005_3FFFh	0014_8000h-0014_FFFFh
10	0001_4000h-0001_5FFFh	0005_4000h-0005_5FFFh	0015_0000h-0015_7FFFh
11	0001_6000h-0001_7FFFh	0005_6000h-0005_7FFFh	0015_8000h-0015_FFFFh
12	0001_8000h-0001_9FFFh	0005_8000h-0005_9FFFh	0016_0000h-0016_7FFFh
13	0001_A000h-0001_BFFFh	0005_A000h-0005_BFFFh	0016_8000h-0016_FFFFh
14	0001_C000h-0001_DFFFh	0005_C000h-0005_DFFFh	0017_0000h-0017_7FFFh
15	0001_E000h-0001_FFFFh	0005_E000h-0005_FFFFh	0017_8000h-0017_FFFFh
16	0002_0000h-0002_1FFFh	0006_0000h-0006_1FFFh	0018_0000h-0018_7FFFh
17	0002_2000h-0002_3FFFh	0006_2000h-0006_3FFFh	0018_8000h-0018_FFFFh
18	0002_4000h-0002_5FFFh	0006_4000h-0006_5FFFh	0019_0000h-0019_7FFFh
19	0002_6000h-0002_7FFFh	0006_6000h-0006_7FFFh	0019_8000h-0019_FFFFh
20	0002_8000h-0002_9FFFh	0006_8000h-0006_9FFFh	001A_0000h-001A_7FFFh
21	0002_A000h-0002_BFFFh	0006_A000h-0006_BFFFh	001A_8000h-001A_FFFFh
22	0002_C000h-0002_DFFFh	0006_C000h-0006_DFFFh	001B_0000h-001B_7FFFh
23	0002_E000h-0002_FFFFh	0006_E000h-0006_FFFFh	001B_8000h-001B_FFFFh
24	0003_0000h-0003_1FFFh	0007_0000h-0007_1FFFh	001C_0000h-001C_7FFFh
25	0003_2000h-0003_3FFFh	0007_2000h-0007_3FFFh	001C_8000h-001C_FFFFh
26	0003_4000h-0003_5FFFh	0007_4000h-0007_5FFFh	001D_0000h-001D_7FFFh
27	0003_6000h-0003_7FFFh	0007_6000h-0007_7FFFh	001D_8000h-001D_FFFFh
28	0003_8000h-0003_9FFFh	0007_8000h-0007_9FFFh	001E_0000h-001E_7FFFh
29	0003_A000h-0003_BFFFh	0007_A000h-0007_BFFFh	001E_8000h-001E_FFFFh
30	0003_C000h-0003_DFFFh	0007_C000h-0007_DFFFh	001F_0000h-001F_7FFFh
31	0003_E000h-0003_FFFFh	0007_E000h-0007_FFFFh	001F_8000h-001F_FFFFh

7.6.2 Программно доступные регистры блока защиты памяти

Регистры блока представлены в карте памяти процессора NMC. Имеются 4 способа записи в каждый регистр:

- Нормальный доступ - устанавливает в регистре записываемое значение;
- Побитовая установка - устанавливает значение 1 в тех разрядах, в которых имеются единицы в записываемом данном, остальные разряды не меняются;
- Побитовый сброс - устанавливает значение 0 в тех разрядах, в которых имеются единицы в записываемом данном, остальные разряды не меняются;
- Обнуление - устанавливает значение 0 во всех битах регистра, независимо от записываемого значения (т. е. имеет значение сам факт записи).

Для каждого способа записи выделен отдельный адрес. Некоторые способы для некоторых регистров недоступны. Чтение регистра может производиться по любому из описнных адресов.

Таблица 7.14 - Программно доступные регистры блока защиты памяти

Адрес	Название	Доступ	Описание
4000_1400h	RD_MASK	ЧТ/ЗП	Маска защищаемых сегментов: 0 - обращение в данный сегмент по чтению разрешено, 1 - обращение в данный сегмент по чтению блокируется.
4000_1402h	RD_MASK_SET	ЗП	Побитовая установка регистра RD_MASK
4000_1404h	RD_MASK_CLEAR	ЗП	Побитовый сброс регистра RD_MASK
4000_1406h	RD_MASK_NULL	ЗП	Обнуление регистра RD_MASK
4000_1408h	RD_IRQ_STATUS_RAW	ЧТ	Статус запросов на прерывание без учёта маски: 0 - чтения в данный сегмент не было, 1 - чтение в данный сегмент было.
4000_140Ah	RD_IRQ_STATUS_RAW_SET	ЗП	Побитовая установка (только для отладки) RD_IRQ_STATUS_RAW
4000_140Ch	RD_IRQ_STATUS_RAW_CLEAR	ЗП	Побитовый сброс регистра RD_IRQ_STATUS_RAW
4000_140Eh	RD_IRQ_STATUS_RAW_NULL	ЗП	Обнуление регистра RD_IRQ_STATUS_RAW
4000_1410h	RD_IRQ_MASK	ЧТ/ЗП	Маска запросов на прерывание от регистра RD_IRQ_STATUS_RAW: 0 - прерывание запрещено, 1 - прерывание разрешено.
4000_1412h	RD_IRQ_MASK_SET	ЗП	Побитовая установка регистра RD_IRQ_MASK
4000_1414h	RD_IRQ_MASK_CLEAR	ЗП	Побитовый сброс регистра RD_IRQ_MASK
4000_1416h	RD_IRQ_MASK_NULL	ЗП	Обнуление регистра RD_IRQ_MASK
4000_1418h	RD_IRQ_STATUS	ЧТ	Статус запросов на прерывание с учётом маски: 0 - нет запроса, 1 - есть запрос.
4000_141Ah	Зарезервировано	-	

Продолжение таблицы 7.14

Адрес	Название	Доступ	Описание
4000_141Ch	Зарезервировано	-	
4000_141Eh	RD_IRQ_STATUS_NULL	ЗП	Обнуление незамаскированных запросов на прерывание (сбрасываются все биты регистров RD_IRQ_STATUS и RD_IRQ_STATUS_RAW, для кото-рых установлен бит RD_IRQ_MASK).
4000_1440h	WR_MASK	ЧТ/ЗП	Маска защищаемых сегментов: 0 - чтение из данного сегмента разрешено, 1 - чтение из данного сегмента запрещено.
4000_1442h	WR_MASK_SET	ЗП	Побитовая установка регистра WR_MASK
4000_1444h	WR_MASK_CLEAR	ЗП	Побитовый сброс регистра WR_MASK
4000_1446h	WR_MASK_NULL	ЗП	Обнуление регистра WR_MASK
4000_1448h	WR_IRQ_STATUS_RAW	ЧТ	Статус запросов на прерывание без учёта маски: 0 - обращений по чтению в данный сегмент не было, 1 - зафиксировано обращение по чтению в данный сегмент.
4000_144Ah	WR_IRQ_STATUS_RAW_SET	ЗП	Побитовая установка WR_IRQ_STATUS_RAW (только для отладки)
4000_144Ch	WR_IRQ_STATUS_RAW_CLEAR	ЗП	Побитовый сброс регистра WR_IRQ_STATUS_RAW
4000_144Eh	WR_IRQ_STATUS_RAW_NULL	ЗП	Обнуление регистра WR_IRQ_STATUS_RAW
4000_1450h	WR_IRQ_MASK	ЧТ/ЗП	Маска запросов на прерывание от регистра WR_IRQ_STATUS_RAW: 0 - прерывание запрещено, 1 - прерывание разрешено.
4000_1452h	WR_IRQ_MASK_SET	ЗП	Побитовая установка регистра WR_IRQ_MASK
4000_1454h	WR_IRQ_MASK_CLEAR	ЗП	Побитовый сброс регистра WR_IRQ_MASK
4000_1456h	WR_IRQ_MASK_NULL	ЗП	Обнуление регистра WR_IRQ_MASK
4000_1458h	WR_IRQ_STATUS	ЧТ	Статус запросов на прерывание с учётом маски: 0 - нет запроса, 1 - есть запрос.
4000_145Ah	Зарезервировано	-	
4000_145Ch	Зарезервировано	-	
4000_145Eh	WR_IRQ_STATUS_NULL	ЗП	Обнуление незамаскированных запросов на прерывание (сбрасываются все биты регистров WR_IRQ_STATUS и WR_IRQ_STATUS_RAW, для которых установлен бит WR_IRQ_MASK).

7.6.3 Прерывания

Каждый экземпляр (чтение и запись) блока защиты памяти имеет один сигнал запроса на прерывание к процессору NMC. Сигнал запроса установлен, пока хотя бы для одного из сегментов RD/WR_IRQ_STATUS_RAW[i]=1 и RD/WR_IRQ_MASK[i]=0. Запрос можно снять программно одним из двух способов:

- С помощью регистра RD/WR_IRQ_CLEAR_RAW - побитовый сброс битов регистра RD/WR_IRQ_STATUS_RAW;
- С помощью регистра RD/WR_IRQ_STATUS_CLEAR – сброс всех незамаскированных битов регистра RD/WR_IRQ_STATUS_RAW.

7.7 Блок управления кэш-памятью команд

Блок управления кэш-памятью команд содержит регистры, с помощью которых производится программная настройка контроллера кэш-памяти команд процессорного ядра NMC4.

Блок содержит два программно доступных регистра:

- Регистр управления;
- Регистр адреса гиперстраницы.

Имеются 3 способа доступа к регистру управления:

- Нормальный доступ - устанавливает в регистре записываемое значение;
- Побитовая установка - устанавливает значение 1 в тех разрядах, в которых имеются единицы в записываемом данном, остальные разряды не меняются;
- Побитовый сброс - устанавливает значение 0 в тех разрядах, в которых имеются единицы в записываемом данном, остальные разряды не меняются.

Для каждого способа записи выделен отдельный адрес. Чтение регистра может производиться по любому из описнных адресов. Регистр адреса гиперстраницы имеет только нормальный способ доступа.

Таблица 7.15 - Программно доступные регистры блока управления кэш-памятью

Адрес	Название	Доступ	Описание
4000_0C00h	CSR	ЧТ/ЗП	Регистр управления кэш-памятью команд
4000_0C02h	CSR_SET	ЧТ/ЗП	Побитовая установка регистра CSR
4000_0C04h	CSR_CLR	ЧТ/ЗП	Побитовый сброс регистра CSR
4000_0C08h	PDA	ЧТ/ЗП	Регистр адреса гиперстраницы

7.7.1 Регистр управления кэш-памятью команд

Таблица 7.16 - Формат регистра управления кэш-памятью команд

Разряды	Название	Доступ	Описание
31-3	-	-	Зарезервировано
2	CRST	ЗП	Запись значения 1 в данное поле сбрасывает содержимое кэш-памяти. При чтении возвращается 0.
1	GTWE	ЧТ/ЗП	Разрешение аппаратной записи в регистр гиперстраницы: 0 - аппаратная запись разрешена, 1 - аппаратная запись запрещена.
0	CEN	ЧТ/ЗП	Разрешение работы кэш-памяти при выборке команд из внешней памяти: 0 - разрешено, 1 - запрещено.

Бит CEN определяет будет ли при выборке команд из внешней памяти анализироваться состояние кэш-памяти. Если работа кэш-памяти запрещена, то команды будут выбираться из внешней памяти минуя кэш-память. Необходимо отметить, что при нулевом значении данного бита состояние кэш-памяти не изменяется, то есть сброс данного поля означает “заморозку” кэш-памяти в состоянии на момент сброса бита CEN.

Бит GTWE разрешает аппаратную запись в регистр адреса гиперстраницы. Если аппаратная запись в регистр GPA разрешена, то при обращении за командой в гиперстраницу, отличную от отраженной в кэш-памяти, произойдет сброс содержимого кэш-памяти и замещение гиперстраницы в регистре GPA. Кэш-память в этом случае будет заполняться с состояния как после системного сброса.

Если аппаратная запись в регистр GPA запрещена, то при обращении за командой в гиперстраницу, отличную от отраженной в кэш-памяти, состояние кэш-памяти не изменится, а команды будут выбираться из внешней памяти так, как будто кэш- память отсутствует.

Данный механизм, в сочетании с программной записью регистра гиперстраницы, позволяет разделить внешнюю память на кэшируемую и не кэшируемую области.

7.7.2 Регистр адреса гиперстраницы

Таблица 7.17 - Формат регистра адреса гиперстраницы

Разряды	Название	Описание
31-20	GyperTag	Поле загрузки адреса гиперстраницы.
19-0	-	Зарезервировано

Регистр предназначен для загрузки адреса гиперстраницы внешней памяти, отражаемой в кэш-памяти. Регистр доступен для записи и чтения. Запись в данный регистр сбрасывает содержимое кэш-памяти и устанавливает записываемое значение как адрес текущей гиперстраницы. Если разрешена аппаратная запись в регистр гиперстраницы, содержимое регистра может меняться, но при чтении возвращается всегда последнее записанное программным способом значение. Если с момента системного сброса записи в регистр не производилось, то возвращается неинициализированное значение.

7.8 Внешние байтовые коммуникационные порты

Синхронные байтовые коммуникационные порты ввода/вывода предназначены для высокоскоростного обмена данными между процессорами или между процессором и внешним устройством по типу “точка - точка”. Микросхема в своем составе содержит четыре коммуникационных порта, по два на каждую процессорную систему. По устройству и функциональным возможностям все порты идентичны.

Основные характеристики каждого из коммуникационных портов:

- Порт должен обеспечивать полудуплексную побайтную передачу 64-разрядных слов;
- Непосредственная коммутация процессоров с помощью восьми линий данных и четырех линий управления;
- Передача в обе стороны с производительностью до 125 Мбайт/сек (при тактовой частоте работы процессора, равной 500 МГц);
- Возможность работы со скоростью 100 и 83,33 Мбайт/сек;
- Синхронный обмен с выставлением строба и данных от передатчика к приёмнику для увеличения скорости обмена, причём данные меняются по каждому фронту переключения строба (по типу Double Data Rate);
- Формирование сигнала готовности от приёмника передатчику для синхронизации их работы;
- Автоматический асинхронный арбитраж шины данных между двумя процессорами.

7.8.1 Внешние выводы коммуникационного порта

В каждый момент времени коммуникационный порт может находиться в одном из двух состояний: в режиме приема или в режиме передачи. В режиме передачи порт является активным устройством, выдающим на шину данные и стробирующие сигналы. В режиме приема порт является пассивным устройством, ожидающим прихода данных. Функциональное назначение выводов коммуникационного порта описано в Таблица 7.18.

Таблица 7.18 - Функциональное описание выводов коммуникационного порта

Обозначение [1)]	**Кол.**	**Тип[3)]**	**Функциональное назначение**	
			Ком. порт в режиме приема	Ком. порт в режиме передачи
CyD	8	I/O	Входы данных	Выходы данных
XCySTRB	1	I/O	Входной строб данных	Выходной строб данных
XCyRDY	1	I/O	Выход готовности к приему	Вход готовности к приему
XCyHOLDI	1	I	Вход разрешения на передачу шины от внешнего устройства	Вход запроса от внешнего устройства на захват шины
XCyHOLDO	1	O	Выход запроса порта на передачу шины	Выход разрешения порта на передачу шины
CyIS	1	I	Состояние коммуникационного порта после системного сброса: 0 - порт в режиме приема, 1 - порт в режиме передачи.	

1) При обозначении выводов для NMPU0 символ “y” для коммуникационного порта CP0 принимает значение 0, а для коммуникационного порта CP1 принимает значение 1, для NMPU1 символ “y” для коммуникационного порта CP2 принимает значение 2, а для коммуникационного порта CP3 принимает значение 3.
2) Для выводов с первым символом «X» в имени активным является низкий уровень сигнала.
3) Используемые обозначения типов выводов:
   I – вход,
   O - выход,
   I/O – двунаправленный вывод.

7.8.2 Организация обмена данными по коммуникационному порту

Перед запуском канала коммуникационного порта необходимо задать начальный адрес и размер пакета передаваемых или принимаемых данных. Значения соответствующих регистров задаются в блоке контроллера ПДП процессорных систем NMPU0 и NMPU1.

Скорость передачи данных на внешнем интерфейсе коммуникационного порта задаётся в системном контроллере процессорной системы. Для этого используется поле RATE регистра управления передающей частью коммуникационного порта (CPC).

После старта канала передачи устройство управления стремится перевести коммуникационный порт в режим передачи. Если порт находится в режиме приема, то запускается процедура арбитража шины коммуникационного порта. После окончания процесса арбитража устройство управления выдает запрос к памяти процессора, переупаковывает считанное из памяти 64-разрядное слово в пакет из восьми байтов и, в случае готовности приемника, выдает пакет на шину коммуникационного порта (CyD[7:0]).

Каждый выдаваемый на шину коммуникационного порта байт сопровождается изменением состояния на выходе XCySTRB. В зависимости от частоты работы коммуникационного порта изменение состояния вывода XCySTRB сдвинуто относительно выдачи байта данных на время, равное одному, двум или трем периодам тактового сигнала процессора. Это позволяет использовать фронты сигнала XCySTRB для фиксации данных на приемном конце.

Сигнал готовности приемника (XCyRDY) фиксируется процессором по положительному фронту сигнала CLK и анализируется при выдаче каждого байта данных.

После старта канала приема, устройство управления находится в режиме ожидания данных с шины коммуникационного порта (CyD[7:0]). Получаемые данные фиксируются процессором как по фронту, так и по срезу сигнала строба данных (XCySTRB). После получения каждых восьми байтов происходит переупаковка данных в 64-разрядное слово и выдается запрос на запись в память процессора.

Если канал приема не готов принять очередной байт данных, то сигнал готовности приемника (XCyRDY) снимается. Передатчик, зафиксировав отсутствие готовности приемника, должен сразу же прекратить выдачу данных и изменение сигнала XCySTRB. Однако прекращение передачи данных при отсутствии готовности приемника не происходит мгновенно после снятия сигнала XCyRDY из-за задержек распространения сигналов от передатчика к приемнику и обратно. Поэтому в канале приема реализован буфер, позволяющий принять данные, отправленные передатчиком до фиксации неготовности приемника. Размер этого буфера накладывает ограничения на время распространения сигналов. Корректная работа канала приема гарантируется, если время распространения сигнала между приемником и передатчиком не более двух периодов тактового сигнала процессора.

Временные диаграммы обмена по коммуникационному порту представлены на

Рисунок 7.8.

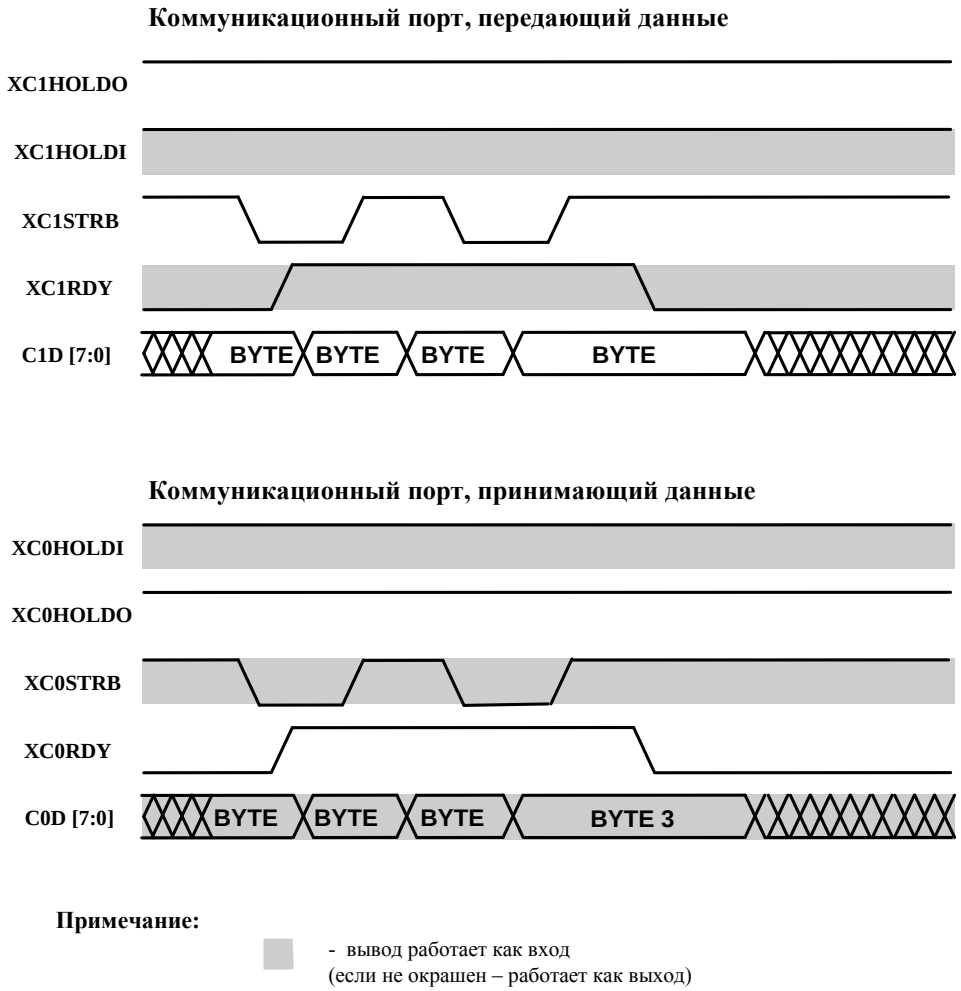


Рисунок 7.8 - Временные диаграммы обмена по коммуникационному порту

7.8.3 Арбитраж шины коммуникационного порта

Процедура арбитража шины позволяет изменить состояние коммуникационного порта с приема на передачу и обратно. Управление процедурой осуществляется с помощью внешних выводов XCyHOLDI и XCyHOLDO. Функциональное назначение этих выводов в различных состояниях коммуникационного порта представлено ранее в Таблица 7.18.

Временные диаграммы арбитража шины представлены ниже на

Рисунок 7.9.

Коммуникационный порт, находящийся в режиме приема, запрашивает шину при запуске канала передачи порта (установке битов EN в регистре CPyRD_CSR) независимо от того, идет в данный момент прием данных или нет. С этой целью на выходе XCyHOLDO устанавливается низкий уровень сигнала.

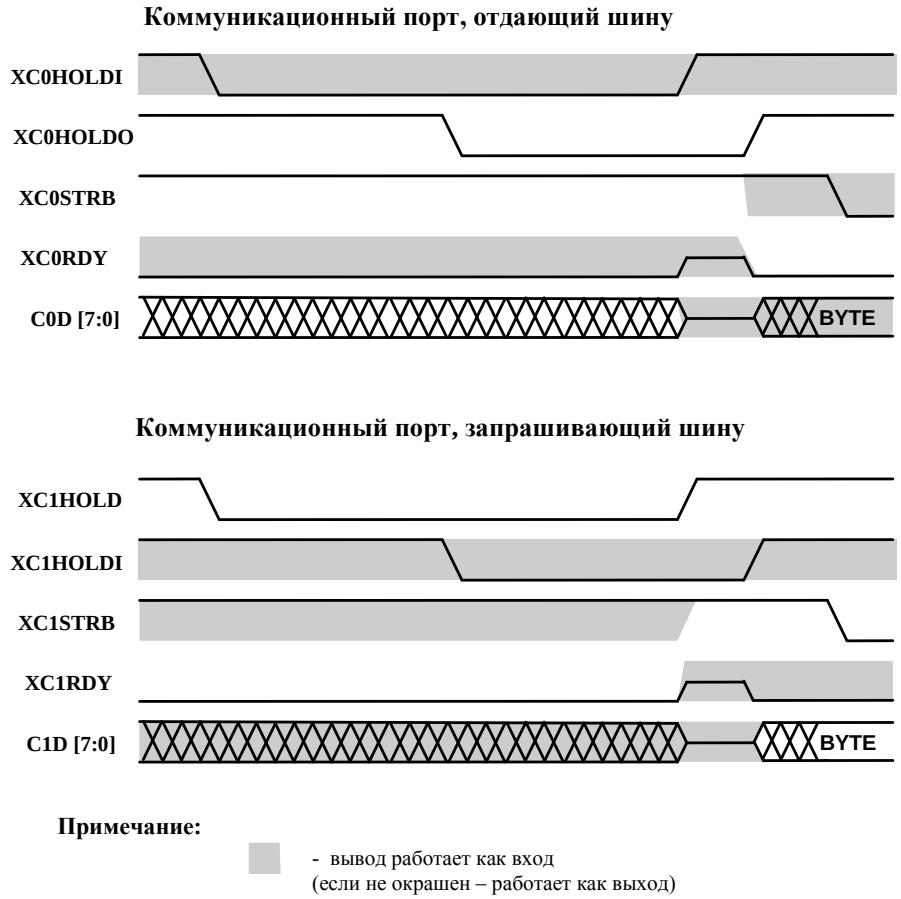


Рисунок 7.9 - Временные диаграммы арбитража шины коммуникационного порта

Если коммуникационный порт, передающий данные, закончил выдачу (бит CPL в регистре CPyRD_CSR установлен) и у него инициализирован канал приема данных (бит EN в регистре CPyWR_CSR установлен), то, получив запрос на захват шины, он выставляет сигнал разрешения на передачу шины.

Для корректного выполнения арбитража шины необходимо, чтобы все переданные данные были записаны портом- приемником в память процессора. Если этого не произойдет, то процедура арбитража останется незаконченной, даже если разрешение на захват шины получено.

После получения сигнала разрешения на захват шины (низкий уровень на входе XCyHOLDI) порт- приемник переводит вывод XCyRDY в состояние “на прием”, вывод XCySTRB в положение “на вывод“ и снимает запрос на захват шины (XCyHOLDO).

Зафиксировав снятие запроса на захват шины, порт-передатчик переводит шину данных CyD и вывод XCyHOLDO в положение “на прием”, вывод XCyHOLDO в состояние “на вывод” и снимает сигнал разрешения на передачу шины.

В свою очередь порт-приемник, зафиксировав снятие разрешения на передачу шины (высокий уровень на входе XCyHOLDI), переводит выводы данных в состояние “на выдачу”. С этого момента он становится передатчиком и начинает осуществлять выдачу данных.

7.9 Контроллер ПДП коммуникационных портов

Контроллер ПДП для коммуникационных портов позволяют осуществлять прямой доступ во внутреннюю память процессорной системы за данными, которые приняты или будут переданы коммуникационными портами.

Контроллер содержит следующие каналы ПДП:

- Канал память-CP0 (CP0TR) – чтение данных для последующей передачи их по коммуникационному порту 0;
- Канал память-CP1 (CP1TR) – чтение данных для последующей передачи их по коммуникационному порту 1;
- Канал память-CP2 (CP2TR) – чтение данных для последующей передачи их по коммуникационному порту 2;
- Канал CP0-память (CP0RC) – запись данных, принятых по коммуникационному порту 0, в память;
- Канал CP1-память (CP1RC) – запись данных, принятых по коммуникационному порту 1, в память;
- Канал CP2-память (CP2RC) – запись данных, принятых по коммуникационному порту 2, в память.

Данные каналы предназначены для обслуживания внешних коммуникационных портов микросхемы, а также для организации канала обмена данными между процессорными системами внутри микросхемы.

В контроллере ПДП все 6 каналов независимы. Передающий и приёмный каналы одного коммуникационного порта могут запускаться независимо, при этом арбитраж шины для коммутационных портов 0 и 1 производится интерфейсным блоком коммуникационного порта.

Каналы ПДП настраиваются с помощью программно доступных регистров. Регистры контроллера расположены в адресном пространстве периферийных устройств процессорных систем, начиная с базового адреса 0x40001800.

Каждый канал управляется своим набором регистров, который отображается в адресном пространстве как выровненный блок из восьми 32-разрядных слов, структура этого набора – общая для всех каналов. Набор регистров канала состоит из:

- Основного счётчика данных (MainCounter);
- Регистра текущего адреса (Address);
- Регистра смещения адреса (Bias);
- Счётчика последовательных данных (RowCounter);
- Регистра режима адресации (AddressMode);
- Регистра управления (Control);
- Регистра масок запросов на прерывание (InterruptMask);
- Регистра состояния (State).

Не указанные в таблице адреса зарезервированы – запись по зарезервированным адресам никак не влияет на устройство, при чтении выдаётся неспецифицированное значение. Неиспользованные старшие разряды имеющихся регистров при чтении возвращают 0. Список регистров приведен в Таблица 7.19.

Таблица 7.19 - Список регистров контроллера ПДП

Название регистра	Адрес в пространстве NMPU0 (NMPU1)	Разрядность	Доступ
Регистры передающего канала коммуникационного порта 0			
CP0TR_MainCounter	0x40001800h	16	ЧТ/ЗП
CP0TR_Address	0x40001802h	32	ЧТ/ЗП
CP0TR_Bias	0x40001804h	32	ЧТ/ЗП
CP0TR_RowCounter	0x40001806h	16	ЧТ/ЗП
CP0TR_AddressMode	0x40001808h	1	ЧТ/ЗП
CP0TR_Control	0x4000180Ah	4	ЧТ/ЗП
CP0TR_InterruptMask	0x4000180Ch	2	ЧТ/ЗП
CP0TR_State	0x4000180Eh	10	ЧТ/ЗП
Регистры приёмного канала коммуникационного порта 0			
CP0RC_MainCounter	0x40001810h	16	ЧТ/ЗП
CP0RC_Address	0x40001812h	32	ЧТ/ЗП
CP0RC_Bias	0x40001814h	32	ЧТ/ЗП
CP0RC_RowCounter	0x40001816h	16	ЧТ/ЗП
CP0RC_AddressMode	0x40001818h	1	ЧТ/ЗП
CP0RC_Control	0x4000181Ah	4	ЧТ/ЗП
CP0RC_InterruptMask	0x4000181Ch	2	ЧТ/ЗП
CP0RC_State	0x4000181Eh	15	ЧТ/ЗП
Регистры передающего канала коммуникационного порта 1			
CP1TR_MainCounter	0x40001C00h	16	ЧТ/ЗП
CP1TR_Address	0x40001C02h	32	ЧТ/ЗП
CP1TR_Bias	0x40001C04h	32	ЧТ/ЗП
CP1TR_RowCounter	0x40001C06h	16	ЧТ/ЗП
CP1TR_AddressMode	0x40001C08h	1	ЧТ/ЗП
CP1TR_Control	0x40001C0Ah	4	ЧТ/ЗП
CP1TR_InterruptMask	0x40001C0Ch	2	ЧТ/ЗП
CP1TR_State	0x40001C0Eh	10	ЧТ/ЗП
Регистры приёмного канала коммуникационного порта 1			
CP1RC_MainCounter	0x40001C10h	16	ЧТ/ЗП
CP1RC_Address	0x40001C12h	32	ЧТ/ЗП
CP1RC_Bias	0x40001C14h	32	ЧТ/ЗП
CP1RC_RowCounter	0x40001C16h	16	ЧТ/ЗП
CP1RC_AddressMode	0x40001C18h	1	ЧТ/ЗП
CP1RC_Control	0x40001C1Ah	4	ЧТ/ЗП
CP1RC_InterruptMask	0x40001C1Ch	2	ЧТ/ЗП
CP1RC_State	0x40001C1Eh	15	ЧТ/ЗП
Регистры передающего канала коммуникационного порта 2			
CP2TR_MainCounter	0x40001800h	16	ЧТ/ЗП
CP2TR_Address	0x40001802h	32	ЧТ/ЗП
CP2TR_Bias	0x40001804h	32	ЧТ/ЗП
CP2TR_RowCounter	0x40001806h	16	ЧТ/ЗП
CP2TR_AddressMode	0x40001808h	1	ЧТ/ЗП
CP2TR_Control	0x4000180Ah	4	ЧТ/ЗП
CP2TR_InterruptMask	0x4000180Ch	2	ЧТ/ЗП
CP2TR_State	0x4000180Eh	10	ЧТ/ЗП
Регистры приёмного канала коммуникационного порта 2			
CP2RC_MainCounter	0x40001810h	16	ЧТ/ЗП
CP2RC_Address	0x40001812h	32	ЧТ/ЗП
CP2RC_Bias	0x40001814h	32	ЧТ/ЗП
CP2RC_RowCounter	0x40001816h	16	ЧТ/ЗП
CP2RC_AddressMode	0x40001818h	1	ЧТ/ЗП
CP2RC_Control	0x4000181Ah	4	ЧТ/ЗП
CP2RC_InterruptMask	0x4000181Ch	2	ЧТ/ЗП
CP2RC_State	0x4000181Eh	15	ЧТ/ЗП

7.9.1 Основной счётчик данных (MainCounter)

Запись в 16 младших разрядов регистра MainCounter устанавливает количество 64-разрядных слов в массиве, который следует предать/принять по коммуникационному порту. При чтении возвращается количество 64-разрядных слов, которое осталось предать/принять.

7.9.2 Регистр текущего адреса (Address)

Запись в регистр Address устанавливает начальный адрес массива, который следует предать/принять по коммуникационному порту. Начальный адрес должен быть выровнен по границе 64-разрядного слова (0-й разряд регистра текущего адреса должен быть равен нулю). При чтении возвращается текущий адрес массива. При задании начального адреса необходимо помнить, что массив данных может располагаться только по адресам «своей» внутренней памяти соответствующей процессорной системы. Для NMPU0 - это адреса в диапазоне от 0h до 3_FFFEh, для NMPU1 – от 0h до 1_FFFEh.

7.9.3 Регистр смещения адреса (Bias)

Регистр Bias используется только при двухмерной адресации, при одномерной адресации он может быть не определён. Запись в него устанавливает смещение адреса при переходе от конца одной строки передаваемого массива до начала следующей. Фактически – это размер в 64-разрядных словах пропускаемого фрагмента.

7.9.4 Счётчик последовательных данных (RowCounter)

Регистр RowCounter используется только при двухмерной адресации, при одномерной адресации он может быть не определён. Запись в его 16 младших разрядов устанавливает размер строки в 64-разрядных словах передаваемого массива.

7.9.5 Регистр режима адресации (AddressMode)

Одноразрядный регистр AddressMode задаёт режим адресации данных в памяти: одномерный или двумерный. Запись в его 0-й разряд нуля определяет одномерную адресацию, единицы - двухмерную.

При одномерной адресации массив данных в памяти расположен непрерывным блоком 64-разрядных данных.

При двухмерной адресации (см. Рисунок 7.10) данный массив находится в памяти не непрерывным блоком, а равноотстоящими друг от друга фрагментами (строками равного размера). Внутри строки адреса данных изменяются последовательно, между строками задаётся расстояние в 64-разрядных словах - Bias.

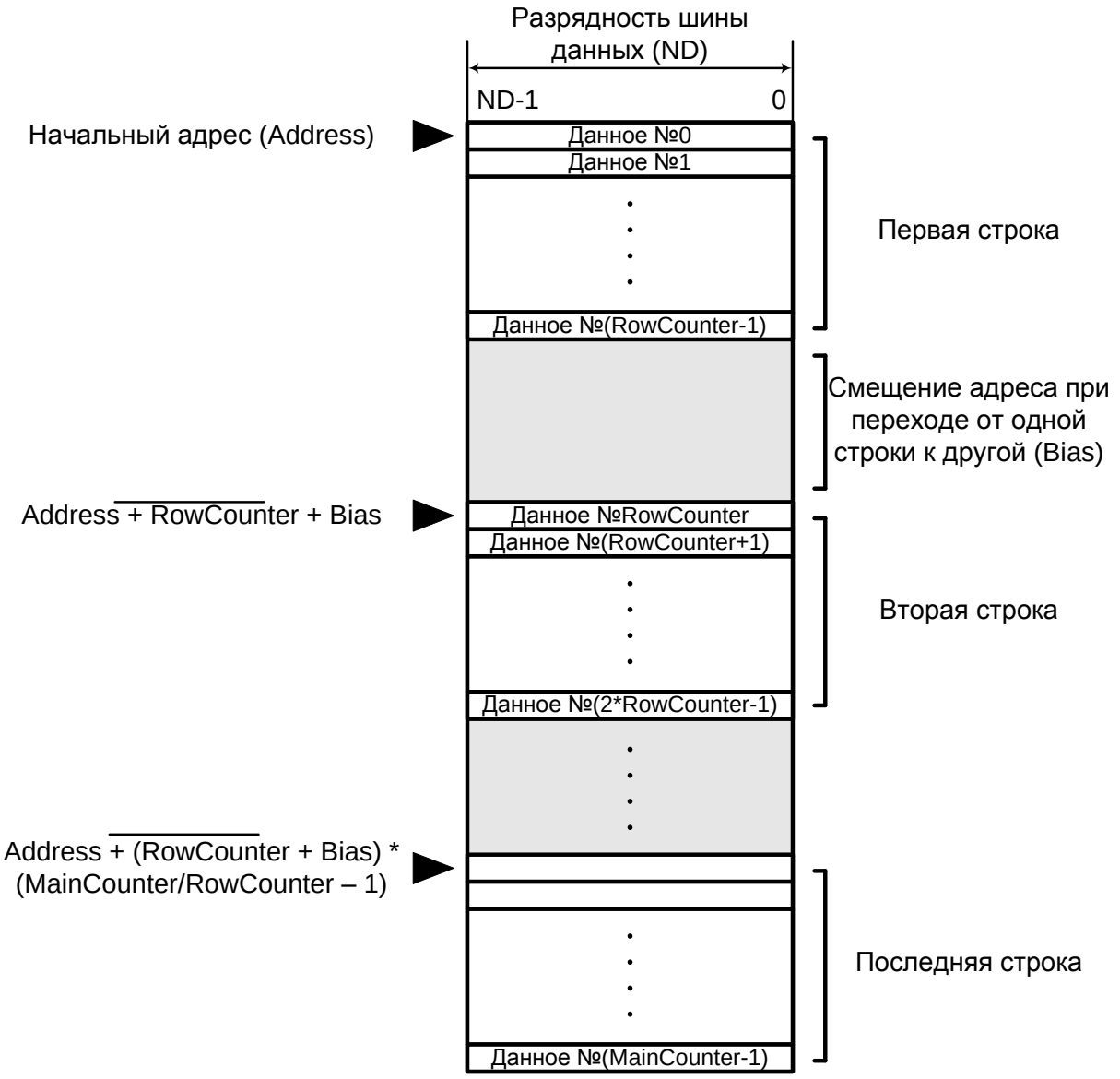


Рисунок 7.10 - Расположение массива данных при двухмерной адресации

7.9.6 Регистр управления (Control)

Формат регистра управления представлен на Рисунок 7.11. Данный регистр используется для управления соответствующим передающим или принимающим каналом коммуникационного порта.

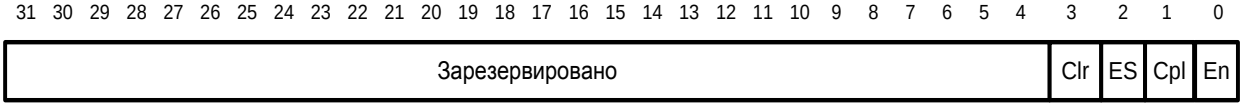


Рисунок 7.11 - Формат регистра управления (Control)

Бит En (0-й разряд). Запись в бит En = 1 запускает передачу/приём.

Бит Cpl (1-й разряд). При чтении Cpl = 1 означает, что текущие передача или приём завершены. Чтобы произвести следующий запуск, необходимо записать Cpl = 0.

Бит ES (2-й разряд). При чтении ES = 1 означает, что передача или приём приостановлены. Бит ES устанавливается аппаратно, если контроллер получил сигнал об ошибке своего обращения в память – по несуществующему или недоступному адресу. Также бит ES можно установить программно. После обработки ошибки бит ES необходимо сбрасывать программно, так как он блокирует дальнейшую работу.

Бит Clr (3-й разряд) показывает, что передающая/принимающая часть контроллера находится в состоянии очистки буфера данных.

7.9.7 Регистр масок запросов на прерывание (InterruptMask)

Формат регистра масок запросов на прерывание представлен на Рисунок 7.12. Данный регистр позволяет запретить выдачу запроса на прерывание по любой из двух причин: нормальное завершение обмена данными или ошибка при обращении в память.

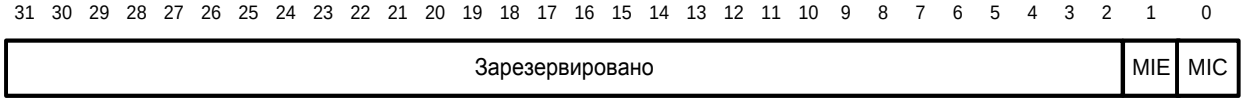


Рисунок 7.12 - Формат регистра масок запросов на прерывание (InterruptMask)

Бит MIC (0-й разряд) – маска запроса на прерывание по нормальному завершению обмена данными:

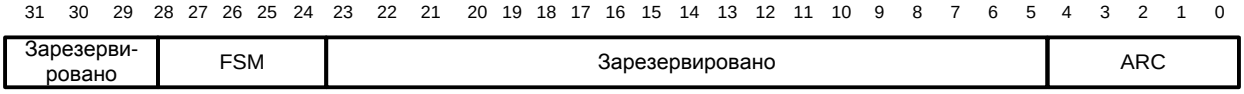
0 – запрос разрешён;
1 – запрос замаскирован.

Бит MIE (1-й разряд) – маска запроса на прерывание по ошибке или программной остановке:

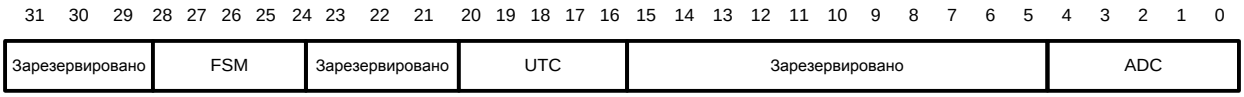
0 – запрос разрешён;
1 – запрос замаскирован.

7.9.8 Регистр состояния (State)

Формат регистров состояния передающего и принимающего канала коммуникационного порта представлен на Рисунок 7.13. Эти регистры доступны только для чтения и могут быть использованы для отладки программ пользователя при обработке ошибочных ситуаций.



а) Формат регистра состояния передающего канала коммуникационного порта



б) Формат регистра состояния принимающего канала коммуникационного порта

Рисунок 7.13 - Формат регистров состояния (State) передающего и принимающего канала коммуникационного порта

Регистр состояния передающего канала коммуникационного порта (см. Рисунок 7.13 а) отражает внутреннее состояние передающей части контроллера:

Поле ARC (разряды 4-0) – счётчик активных запросов. Показывает количество данных, запрос на чтение которых уже отдан, но данные ещё не отправлены на коммуникационную шину. При приостановке передающей части сумма MainCounter + ARC показывает реальное количество "недоотправленных" на коммуникационную шину данных.

Поле FSM (разряды 28-24) – показывает текущее состояние конечного автомата передающей части:

01h – Idle – бездействие,
02h – ReadSend – чтение из памяти и отправка,
04h – SendOnly – чтение завершено, только отправка,
08h – Complete – отправка завершена,
1Ch – DataMiss – режим очистки буфера данных.

Регистр состояния принимающего канала коммуникационного порта (см. Рисунок 7.13 б) показывает внутреннее состояние приёмной части блока коммуникационного порта:

Поле ADC (разряды 4-0) – счётчик доступных данных. Показывает количество данных, имеющихся в приёмном буфере блока коммуникационного порта, но ещё не выданных на запись в память.

Поле UTC (разряды 20-16) – счётчик незавершённых транзакций. Показывает количество транзакций записи на шине AXI3, для которых ещё не получено подтверждения.

Поле FSM (разряды 28-24) – показывает текущее состояние конечного автомата передающей части.

01h – Idle – бездействие,
02h – ReceiveWrite – приём и запись данных,
04h – UncompleteWrite – приём завершён, только запись,
08h – Complete – запись завершена,
1Ch – DataMiss – режим очистки буфера данных.

7.9.9 Прерывания от коммуникационных портов

Коммуникационные порты могут формировать до шести запросов на прерывание своей процессорной системе - NMPU0 или NMPU1:

- Запрос на прерывание по завершении работы передающего канала коммуникационного порта 0;
- Запрос на прерывание по завершении работы передающего канала коммуникационного порта 1;
- Запрос на прерывание по завершении работы передающего канала коммуникационного порта 2;
- Запрос на прерывание по завершении работы приёмного канала коммуникационного порта 0;
- Запрос на прерывание по завершении работы приёмного канала коммуникационного порта 1;
- Запрос на прерывание по завершении работы приёмного канала коммуникационного порта 2.

Запрос на прерывание по завершении работы передающего канала коммуникационного порта i (i = 0, 1, 2) формируется, когда выполняются одно из условий:

- Установлен бит Cpl регистра CPiTR_Control и сброшен бит маски MIC в регистре CPiTR_InterruptMask (в т.ч. в конце процесса передачи);
- Установлен бит ES регистра CPiTR_Control и сброшен бит маски MIE в регистре CPiTR_InterruptMask.

Запрос на прерывание по завершении работы принимающего канала коммуникационного порта i (i = 0, 1, 2) формируется, когда выполняются одно из условий:

- Установлен бит Cpl регистра CPiRC_Control и сброшен бит маски MIC в регистре CPiRC_InterruptMask (в т. ч. в конце процесса передачи);

- Установлен бит ES регистра CPiRC_Control и сброшен бит маски MIE в регистре CPiRC_InterruptMask.

7.10 Контроллер прерываний процессорной системы

Контроллер прерываний процессорной системы обрабатывает сигналы запросов на прерывание, направленные от устройств системы (как расположенных внутри неё, так и внешних) к процессорному ядру NMC4. Функции контроллера прерываний заключаются в следующем:

- Детектирование запросов,
- Маскирование неиспользуемых запросов,
- Арбитраж активных запросов,
- Формирование адреса-вектора перехода,
- Подтверждение или разрешение следующих запросов после обработанного.

Структурная схема контроллера прерываний представлена на рисунке 7.14. Контроллер имеет 40 входных сигналов запросов. Поступающий запрос фиксируется в регистре запросов IRR. Далее запрос может быть маскирован регистром IMR. Схема приоритетов (Priority Unit) выбирает запрос с наибольшим приоритетом (схема описана ниже), формирует запрос на прерывание для ядра NMC4 и соответствующий ему адрес-вектор прерывания.

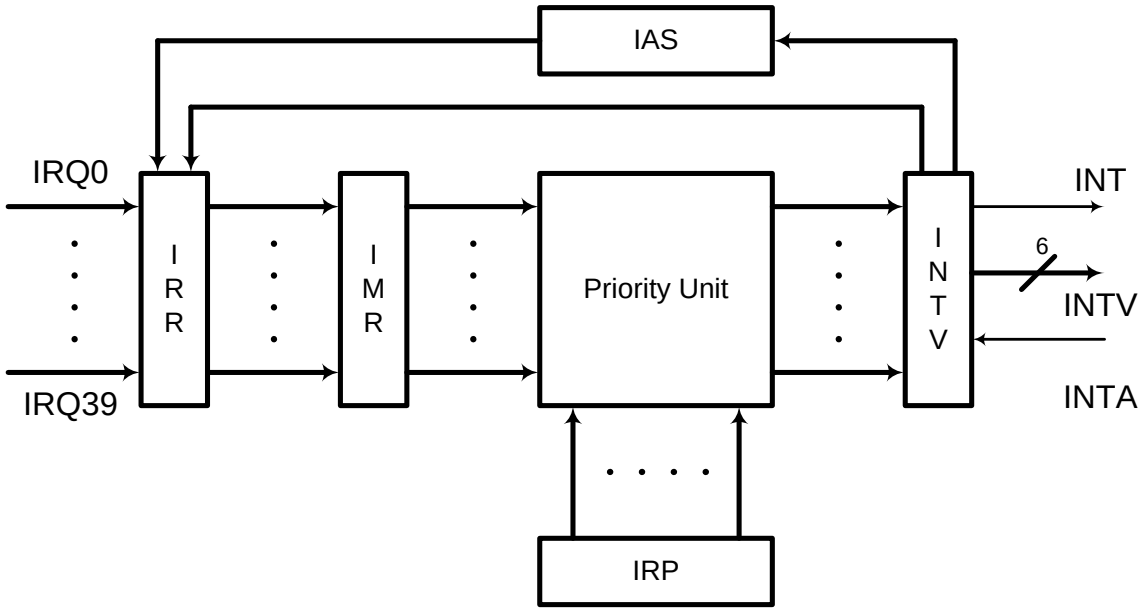


Рисунок 7.14 - Структурная схема контроллера прерываний

Если сигнал INTA имеет активный уровень, то считается, что процессорное ядро NMC4 зафиксировало прерывание. При этом бит в регистре IRR сбрасывается, а также, если данный запрос работает в режиме программного подтверждения, аппаратно устанавливается соответствующий бит в регистре IAS. Установка некоторого разряда IAS блокирует обработку новых прерываний, поступающих по той же линии IRQ. Сброс разрядов IAS выполняется программно с помощью команды записи процессорного ядра NMC4.

Если сигнал INTA имеет неактивный уровень, то запрос на прерывание INT будет стоять до прихода сигнала INTA. При этом адрес-вектор INTV на выходе блока INTC не

фиксируется, т. е. если на вход IRQx поступит более приоритетное прерывание, то адрес-вектор изменится и будет соответствовать более приоритетному прерыванию.

7.10.1 Прерывания процессорной системы

В Таблица 7.20 приведён список всех прерываний процессорной системы, причём выделены все прерывания от блоков, входящих в её состав.

Для процессорной системы NMPU1 прерывания с номерами 0-5 всегда равны нулю, поскольку они формируются сопроцессором арифметики с плавающей точкой (данный сопроцессор имеется только в системе NMPU0).

Таблица 7.20 - Прерывания процессорной системы

№	Поз. бита	Описание	Режим подтв.	Адрес-вектор
0	0	прерывание 0 от сопроцессора арифметики с плавающей точкой (некорректные данные)	А	0x0000_0020 hex-
1	1	прерывание 1 от сопроцессора арифметики с плавающей точкой (переполнение)	А	0x0000_0028 hex-
2	2	прерывание 2 от сопроцессора арифметики с плавающей точкой (потеря значимости)	А	0x0000_0030 hex-
3	3	прерывание 3 от сопроцессора арифметики с плавающей точкой (потеря точности)	А	0x0000_0038 hex-
4	4	прерывание 4 от сопроцессора арифметики с плавающей точкой (потеря данного)	А	0x0000_0040 hex-
5	5	прерывание 5 от сопроцессора арифметики с плавающей точкой (неправильная команда)	А	0x0000_0048 hex
6	6	прерывание от системного интегратора (обращение блока выборки команд в периферийную область)	А	0x0000_0050 hex
7	7	прерывание от моста «системный интегратор – AXI» (ошибка обращения во внешнюю память)	П	0x0000_0058 hex
8	8	прерывание от блока защиты памяти (защита по записи)	П	0x0000_0060 hex
9	9	прерывание от блока защиты памяти (защита по чтению)	П	0x0000_0068 hex
10	10	прерывание от блока таймеров (таймер 0)	А	0x0000_0070 hex
11	11	прерывание от блока таймеров (таймер 1)	А	0x0000_0078 hex
12	12	прерывание 0 от системного контроллера (межпроцессорное прерывание)	А	0x0000_0080 hex
13	13	прерывание 1 от системного контроллера (межпроцессорное прерывание)	А	0x0000_0088 hex
14	14	прерывание 2 от системного контроллера (межпроцессорное прерывание)	А	0x0000_0090 hex
15	15	прерывание 3 от системного контроллера (межпроцессорное прерывание)	А	0x0000_0098 hex
16	16	прерывание от блока сторожевого таймера WDT	П	0x0000_00A0 hex
17	17	прерывание 0 от блока интервальных таймеров (DIT)	П	0x0000_00A8 hex
18	18	внешнее прерывание 0	П	0x0000_00B0 hex
19	19	внешнее прерывание 1	П	0x0000_00B8 hex
20	20	прерывание от контроллера USB	П	0x0000_00C0 hex
21	21	прерывание от контроллера SPI	П	0x0000_00C8 hex
22	22	прерывание от контроллера внешней памяти (EMIC)	П	0x0000_00D0 hex
23	23	прерывание 1 от блока интервальных таймеров (DIT)	П	0x0000_00D8 hex
24	24	прерывание от коммуникационного порта CP0 (передающий канал)	А	0x0000_00E0 hex

Продолжение таблицы 7.20

№	Поз. бита	Описание	Режим подтв.	Адрес-вектор
25	25	прерывание от коммуникационного порта CP0 (принимающий канал)	А	0x0000_00E8 hex
26	26	прерывание от коммуникационного порта CP1 (передающий канал)	А	0x0000_00F0 hex
27	27	прерывание от коммуникационного порта CP1 (принимающий канал)	А	0x0000_00F8 hex
28	28	прерывание от коммуникационного порта CP2 (передающий канал)	А	0x0000_0100 hex
29	29	прерывание от коммуникационного порта CP2 (принимающий канал)	А	0x0000_0108 hex
30	30	внешнее прерывание 2	П	0x0000_0110 hex
31	31	внешнее прерывание 3	П	0x0000_0118 hex
32	0	прерывание по завершении работы канала ПДП память-память (от MDMAC)	П	0x0000_0120 hex
33	1	прерывание по ошибке доступа в память канала ПДП память-память (от MDMAC)	П	0x0000_0128 hex
34	2	прерывание по завершении работы канала ПДП память-периферия (от KDMAC)	П	0x0000_0130 hex
35	3	прерывание по завершении работы канала ПДП периферия-память (от KDMAC)	П	0x0000_0138 hex
36	4	прерывание 4 от системного контроллера (межпроцессорное прерывание)	А	0x0000_0140 hex
37	5	прерывание 5 от системного контроллера (межпроцессорное прерывание)	А	0x0000_0148 hex
38	6	прерывание 6 от системного контроллера (межпроцессорное прерывание)	А	0x0000_0150 hex
39	7	прерывание 7 от системного контроллера (межпроцессорное прерывание)	А	0x0000_0158 hex

7.10.2 Режимы подтверждения запросов

Контроллер прерываний процессорной системы поддерживает 2 режима обработки запросов на прерывание от устройств системы:

- Режим аппаратного подтверждения;
- Режим программного подтверждения.

Режим для каждого из устройств выбран при разработке и не может быть изменён.

В режиме аппаратного подтверждения контроллер фиксирует запрос в регистре IRR в момент положительного фронта входного сигнала запроса. После того как ядро принимает запрос, контроллер готов принять следующий запрос (следующий положительный фронт сигнала запроса) по этому же входному сигналу.

В режиме программного подтверждения контроллер фиксирует запрос, когда обнаруживает высокий уровень входного сигнала запроса, если не установлен соответствующий бит в регистре статуса IAS. В момент, когда ядро NMC принимает адрес-вектор прерывания, бит регистра IRR сбрасывается, а бит регистра IAS устанавливается. Снятие бита в регистре IAS в этом режиме производится программно – команда снятия бита регистра IAS подтверждает обработку текущего запроса и разрешает генерацию следующего.

При работе с устройствами, которые работают в режиме программного подтверждения следует обеспечить, чтобы команда снятия запроса в запрашивающем устройстве выполнялась строго раньше, чем команда снятия бита статуса прерывания (IAS) в контроллере прерываний. Рекомендуется следующий алгоритм программного подтверждения:

1) запись в регистр снятия запроса в запрашивающем устройстве (в этот момент устройство снимет запрос),
2) чтение из любого регистра запрашивающего устройства (чтение выполнится строго после записи),
3) арифметическая или логическая операция с прочитанным значеним (выполнится строго после чтения),
4) запись результата операции в любой зарезервированный регистр контроллера прерываний (выполнится строго после операции в АЛУ),
5) запись значения 1 в активный бит регистра IAS (эта запись выполнится последней, данный бит сбросится и контроллер будет готов принять следующий запрос от соответствующего устройства).

7.10.3 Приоритеты запросов

Каждому входному сигналу запроса можно присвоить уровень приоритета: высокий или низкий. Арбитраж производится сначала между всеми запросами с высоким приоритетом, затем, если нет запросов с высоким приоритетом, между всеми запросами с низким. Для сигналов с одинаковым уровнем приоритета имеет значение порядковый номер сигнала: наибольший приоритет имеет 0-й запрос, следующий приоритет у 1-го запроса и так далее до 39-го запроса, который имеет наименьший приоритет.

7.10.4 Программно доступные регистры контроллера прерываний

Все регистры контроллера прерываний 32-разрядные, каждый бит регистра соответствует одному входному сигналу запроса. Имеется 2 набора регистров:

- Для запросов с номерами от 0 до 31 (IRRL, IMRL, IAML, IRPL, IASL);
- Для запросов с номерами от 32 до 39 (IRRH, IMRH, IAMH, IRPH, IASH). В этой группе регистров используются только 8 младших разрядов (с 0-го по 7-й), остальные разряды зарезервированы для будущих применений.

Регистры блока представлены в карте памяти процессора NMC. Имеются 4 способа записи в каждый регистр:

- Нормальный доступ - устанавливает в регистре записываемое значение;
- Побитовая установка - устанавливает значение 1 в тех разрядах, в которых имеются единицы в записываемом данном, остальные разряды не меняются;
- Побитовый сброс - устанавливает значение 0 в тех разрядах, в которых имеются единицы в записываемом данном, остальные разряды не меняются;
- Обнуление - устанавливает значение 0 во всех битах регистра, независимо от записываемого значения (т. е. эффект производит сам факт записи по выделенному адресу).

Для каждого способа записи выделен отдельный адрес. Некоторые способы для некоторых регистров недоступны. Чтение регистра может производиться по любому из описнных адресов.

Таблица 7.21 - Программно доступные регистры контроллера прерываний

Адрес	Название	Доступ	Описание
4000_0400h	IRRL	ЧТ	Регистр запросов на прерывание с номерами от 0 до 31. Значение 1 в бите данного регистра показывает наличие запроса, для которого ещё не выполнена команда перехода по адресу-вектору прерывания. Программный сброс битов данного регистра производится только с помощью регистра IRRL_CLR.
4000_0402h	IRRL_SET	ЧТ/ЗП	Побитовая установка регистра IRRL.
4000_0404h	IRRL_CLR	ЧТ/ЗП	Побитовый сброс регистра IRRL. Программный сброс бита регистра IRRL следует делать, когда данный запрос замаскирован.
4000_0406h	-	-	Зарезервировано.
4000_0408h	IMRL	ЧТ/ЗП	Регистр маски прерываний с номерами о 0 до 31: 0 - прерывание запрещено, 1 - прерывание разрешено.
4000_040Ah	IMRL_SET	ЧТ/ЗП	Побитовая установка регистра IMRL.
4000_040Ch	IMRL_CLR	ЧТ/ЗП	Побитовый сброс регистра IMRL.
4000_040Eh	IMRL_NULL	ЧТ/ЗП	Сброс всех битов регистра IMRL.
4000_0410h	IRPL	ЧТ/ЗП	Регистров приоритетов прерываний с номерами от 0 до 31: 0 – высокий приоритет, 1 – низкий приоритет.
4000_0412h	IRPL_SET	ЧТ/ЗП	Побитовая установка регистра IRPL.
4000_0414h	IRPL_CLR	ЧТ/ЗП	Побитовый сброс регистра IRPL.
4000_0416h	IRPL_NULL	ЧТ/ЗП	Сброс всех битов регистра IRPL.
4000_0418h	IASL	ЧТ	Регистр подтверждения и статуса запросов с номерами от 0 до 31: 0 - запрос обработан, 1 - запрос обрабатывается (выполнен или выполняется переход по адресу-вектору, но бит регистра IASL ещё не сброшен).
4000_041Ah	-	-	Зарезервировано.
4000_041Ch	IASL_CLR	ЧТ/ЗП	Побитовый сброс регистра IASL.
4000_041Eh	-	-	Зарезервировано.

Продолжение таблицы 7.21

Адрес	Название	Доступ	Описание
4000_0440h	IRRH	ЧТ	Регистр запросов на прерывание с номерами от 32 до 39. Значение 1 в бите данного регистра показывает наличие запроса, для которого ещё не выполнена команда перехода по адресу-вектору прерывания. Программный сброс битов данного регистра производится только с помощью регистра IRRH_CLR.
4000_0442h	IRRH_SET	ЧТ/ЗП	Побитовая установка регистра IRRH.
4000_0444h	IRRH_CLR	ЧТ/ЗП	Побитовый сброс регистра IRRH. Программный сброс бита регистра IRRH следует делать, когда данный запрос замаскирован.
4000_0446h	-	-	Зарезервировано.
4000_0448h	IMRH	ЧТ/ЗП	Регистр маски прерываний с номерами о 0 до 31: 0 - прерывание запрещено, 1 - прерывание разрешено.
4000_044Ah	IMRH_SET	ЧТ/ЗП	Побитовая установка регистра IMRH.
4000_044Ch	IMRH_CLR	ЧТ/ЗП	Побитовый сброс регистра IMRH.
4000_044Eh	IMRH_NULL	ЧТ/ЗП	Сброс всех битов регистра IMRH.
4000_0450h	IRPH	ЧТ/ЗП	Регистров приоритетов прерываний с номерами от 32 до 39: 0 – высокий приоритет, 1 – низкий приоритет.
4000_0452h	IRPH_SET	ЧТ/ЗП	Побитовая установка регистра IRPH.
4000_0454h	IRPH_CLR	ЧТ/ЗП	Побитовый сброс регистра IRPH.
4000_0456h	IRPH_NULL	ЧТ/ЗП	Сброс всех битов регистра IRPH.
4000_0458h	IASH	ЧТ	Регистр подтверждения и статуса запросов с номерами от 32 до 39: 0 - запрос обработан, 1 - запрос обрабатывается (выполнен или выполняется переход по адресу-вектору, но бит регистра IASH ещё не сброшен).
4000_045Ah	-	-	Зарезервировано.
4000_045Ch	IASH_CLR	ЧТ/ЗП	Побитовый сброс регистра IASH.
4000_045Eh	-	-	Зарезервировано.

7.10.4.1 Регистры запросов на прерывание (IRRL, IRRH)

Каждый i-й разряд регистра IRRL соответствует i-му запросу на прерывание, а i-й разряд регистра IRRH – (i+32)-му запросу. При чтении каждый бит показывает состояние запроса: 0 – запрос не активен или уже удовлетворён, 1 – запрос активен. Запись значения 0 в бит сбрасывает этот бит.

Бит регистра IRRL(IRRH) устанавливается аппаратно в момент приёма запроса, снимается также аппаратно в момент, когда ядро принимает адрес-вектор данного прерывания. Бит может установиться аппаратно, даже если данный запрос замаскирован - в этом случае перед снятием маски следует выполнить программный сброс бита путём записи значения 1.

Функция побитовой установки регистра IRRL(IRRH) добавлена для упрощения отладки программного обеспечения.

7.10.4.2 Регистры маски прерываний (IMRL, IMRH)

Каждый i-й разряд регистра IMRL соответствует i-му запросу на прерывание, а i-й разряд регистра IMRH – (i+32)-му запросу. Значения битов: 0 – запрос маскирован, 1 – запрос разрешён.

Регистр маски прерываний разрешает или запрещает прерывание по отдельным сигналам запроса. Если бит регистра IMRL(IMRH) имеет значение 0, то соответствующее прерывание запрещено, если значение 1 - то разрешено. Значение регистра допускается изменять в любой момент времени. При снятии маски следует иметь в виду, что разрешаемый запрос может быть активен, то есть установлено значение 1 в бите регистра IRRL(IRRH). Если такой запрос не нужно обрабатывать, то перед снятием маски следует снять активный бит регистра IRRL(IRRH) путём записи в этот бит значения 1.

7.10.4.3 Регистры приоритетов (IPRL, IPRH)

Каждый i-й разряд регистра IPRL соответствует i-му запросу на прерывание, а i-й разряд регистра IPRH – (i+32)-му запросу. Значения битов: 0 – высокий уровень приоритета, 1 – низкий уровень приоритета

Регистр приоритетов устанавливает уровень приоритета для каждого сигнала запроса. Допускается менять значение этого регистра в любой момент времени.

7.10.4.4 Регистры подтверждения и статуса запросов (IASL, IASH)

Каждый i-й разряд регистра IASL соответствует i-му запросу на прерывание, а i-й разряд регистра IASH – (i+32)-му запросу. При чтении показывает состояние запроса от периферийного устройства: 0 - запрос обработан, 1 - запрос обрабатывается (выполнен или выполняется переход по адресу-вектору, но бит регистра IASL ещё не сброшен).

Регистр подтверждения и статуса используется только для запросов, работающих в режиме программного подтверждения. Бит в регистре IASL(IASH) устанавливается аппаратно в момент, когда ядро принимает адрес-вектор данного прерывания. Сброс бита производится программно с помощью команды побитового сброса (IASL_CLR или IASH_CLR). Если бит регистра IASL(IASH) установлен, аппаратная установка соответствующего бита в регистре IRRL(IRRH) блокируется контроллером. Сброс бита регистра IASL(IASH) разрешает следующий запрос по той же линии. Для запросов, работающих в режиме аппаратного подтверждения, биты регистра IASL(IASH) всегда имеют значение 0.

8 Периферийные устройства микросхемы 1879ВМ6Я

8.1 Общие сведения

Периферийные устройства процессора служат для фиксации внешних событий, организации ввода-вывода данных, тестирования процессора и выполнения процедуры начальной инициализации. Обе процессорные системы NMPU0 и NMPU1 имеют программный доступ к регистрам этих периферийных устройств как на запись, так и на чтение. Большинство периферийных устройств выполнены в виде IP-блоков и имеют стандартные интерфейсы в соответствии со спецификацией шины AMBA компании ARM Limited. В состав периферийных устройств процессора входят следующие блоки:

- Высокоскоростные периферийные устройства – данные устройства подключены к процессорному коммутатору AMBA AXI спецификации 3.0:
  EMI – контроллер внешней динамической памяти (контроллер DDR2).
  MDMAC – контроллер ПДП. Под управлением контроллера осуществляется пересылка данных между банками внутренней памяти и внешней памятью;
  SHMEM –разделяемая память, состоящая из двух независимых банков памяти типа SRAM с организацией 32К х 64 разряда каждый.
  USB 2.0 – контроллер интерфейса USB версии 2.0 в конфигурации device;
  BROMC – контроллер начальной загрузки с встроенным ПЗУ. После перевода сигнала системного сброса в неактивное состояние контроллер осуществляет загрузку программы начальной инициализации во внутреннюю память одной из процессорных систем и разрешает выборку команд соответствующему процессорному ядру;
  KDMAC – контроллер ПДП. Под управлением данного контроллера осуществляется пересылка данных между банками внутренней памяти и периферийными устройствами.
- Низкоскоростные периферийные устройства – данные устройства подключены к периферийной шине AMBA APB спецификации 2.0 Шина AMBA APB подключена к процессорному коммутатору AMBA AXI через стандартный мост AXI/APB Bridge:
  WDT – сторожевой таймер.
  DIT – блок интервальных таймеров.
  EXTIRC – контроллер внешних прерываний.
  GPIO – блок программируемых входов/выходов общего назначения.
  SPI – контроллер синхронного последовательного интерфейса. Контроллер обеспечивает подключение до четырех устройств с внешним интерфейсом Motorola SPI, Texas Instruments SPI или National Semiconductors Microwire.
  SC – системный контроллер, управляющий выбором режима начальной загрузки.

8.2 Контроллер интерфейса с внешней памятью EMI

Контроллер интерфейса с внешней памятью предназначен для организации и управления обменом данными междупроцессорными системами и микросхемами памяти, подключенными к внешней шине микросхемы. Контроллер имеет следующие характеристики:

- Тип поддерживаемой памяти DDR2 SDRAM;
- Частота шины 400 МГц;
- Разрядность внешней шины данных, бит – 32;
- Количество сигналов выбора микросхемы (chip select) – 2;
- Количество банков в микросхеме памяти – 8.

Начальная инициализация, управление контроллера EMI осуществляется по шине AMBA AXI. Конфигурационные регистры располагаются в адресном пространстве процессорных ядер по адресам 0x1002_0000h – 1002_03FFh.

Обращения к внешним микросхемам памяти SDRAM осуществляется по шине AMBA AXI. Адресное пространство в адресном пространстве процессорных ядер для обращения к микросхемам внешней памяти 0x2000_0000h – 0x3FFF_FFFFh. Декодирование адреса на шине AMBA AXI контроллером EMI в сигнал выбора микросхем памяти (XCSi), адрес столбца, адрес строки и номер банка внешней памяти происходит автоматически. Параметры декодирования настраиваются в конфигурационных регистрах EMI.

Перед тем как производить обращения к внешней памяти, необходимо инициализировать контроллер EMI. Процесс инициализации состоит из двух частей: инициализация DLL и программная настройка контроллера. Инициализация DLL осуществляется при помощи системного контроллера SC. Программная настройка контроллера включает следующие действия:

- Настройку конфигурационных регистров контроллера EMI;
- Запись управляющего регистра микросхемы внешней памяти;
- Перевод контроллера EMI в активное состояние.

Пример программы инициализации контроллера EMI приведён в приложении А. Поскольку требуется настроить более сотни конфигурационных регистров, причём делается это один раз, и больше программист к этим регистрам не обращается, их описание не приводится с целью не загромождать данное руководство.

В Таблица 8.1 приведен список внешних выводов микросхемы, относящихся к контроллеру EMI.

Таблица 8.1 - Выводы микросхемы, входящие в состав контроллера EMI

Вывод	Тип буфера	Примечание
DQ0 … DQ31	I/O	Шина данных
A0 … A14	O	Шина адреса
XCS0, XCS1	O	Выборка банков внешней памяти
XWE	O	Разрешение записи в память
XRAS	O	Строб адреса строки
XCAS	O	Строб адреса столбца
DM0…DM3	I/O	Сигналы маскирования данных при записи
DQS0…DQS3	I/O	Сигналы стробов данных
XDQS0…XDQS3	I/O	Инверсные сигналы стробов данных
BA0…BA2	O	Сигналы выбора банка памяти
ODT0, ODT1	O	Сигналы ODT (Memory on-die termination control signal)
CK0, CK1	O	Синхросигналы шины
XCK0, XCK1	O	Инверсные синхросигналы шины
CKE0, CKE1	O	Сигналы разрешения синхросигналов
VREF1… VREF3	AI	Входы напряжения смещения

8.3 Контроллер ПДП память-память MDMAC

Контроллер обеспечивает обмен данными между разными блоками памяти: внутренней (адреса от 0x0004_0000h до 0x0009_FFFFh), общей (адреса от 0x000A_0000h до 0x000B_FFFFh) и внешней (адреса от 0x2000_0000h до 0x3FFF_FFFFh). Особенностями контроллера ПДП являются:

- Наличие одного универсального канала, который состоит из двух подканалов – передающего и принимающего. Эти каналы настраиваются и запускаются программно, а останавливаются либо аппаратно (по окончании обмена или по ошибке), либо программно;
- Встроенный буфер на 32 64-разрядных слова (256 байт);
- Аппаратная поддержка пакетов до 16 64-разрядных слов на шине AMBA AXI с целью более эффективного её использования.

Передача данных реализуется контроллером с помощью транзакций чтения и записи на шине AMBA AXI: контроллер производит чтение данных из источника с помощью передающего подканала, помещает данные во встроенный буфер, затем записывает данные в устройство-приёмник с помощью принимающего подканала.

8.3.1 Программно доступные регистры контроллера MDMAC

Каналы ПДП настраиваются с помощью программно доступных регистров передающего и приёмного подканалов. Регистры контроллера расположены в адресном пространстве периферийных устройств процессорных систем, начиная с базового адреса 0x1001_0000h.

Каждый подканал управляется своим набором регистров, который отображается в адресном пространстве как выровненный блок из восьми 32-разрядных слов, структура этого набора – общая для всех каналов. Набор регистров канала состоит из:

- Основного счётчика данных (MainCounter);
- Регистра текущего адреса (Address);
- Регистра смещения адреса (Bias);
- Счётчика последовательных данных (RowCounter);
- Регистра режима адресации (AddressMode);
- Регистра управления (Control);
- Регистра масок запросов на прерывание (InterruptMask);
- Регистра состояния (State).

Перечисленные выше регистры как по разрядности, так и по составу функциональных полей совпадают с регистрами передающего и приёмного каналов коммуникационных портов, поэтому их описание см. в п. 7.9.

Список регистров приведен в Таблица 8.2. Не указанные в таблице адреса зарезервированы – запись по зарезервированным адресам никак не влияет на устройство, при чтении выдаётся неспецифицированное значение. Неиспользованные старшие разряды имеющихся регистров при чтении возвращают 0.

Таблица 8.2 - Список регистров контроллера MDMAC

Название регистра	Адрес в пространстве NMPU0 (NMPU1)	Разрядность	Доступ
Регистры передающего подканала MDMAC			
DMATR_MainCounter	0x1001_0000h	16	ЧТ/ЗП
DMATR_Address	0x1001_0002h	32	ЧТ/ЗП
DMATR_Bias	0x1001_0004h	32	ЧТ/ЗП
DMATR_RowCounter	0x1001_0006h	16	ЧТ/ЗП
DMATR_AddressMode	0x1001_0008h	1	ЧТ/ЗП
DMATR_Control	0x1001_000Ah	4	ЧТ/ЗП
DMATR_InterruptMask	0x1001_000Ch	2	ЧТ/ЗП
DMATR_State	0x1001_000Eh	10	ЧТ/ЗП
Регистры приёмного подканала MDMAC			
DMARC_MainCounter	0x1001_0010h	16	ЧТ/ЗП
DMARC_Address	0x1001_0012h	32	ЧТ/ЗП
DMARC_Bias	0x1001_0014h	32	ЧТ/ЗП
DMARC_RowCounter	0x1001_0016h	16	ЧТ/ЗП
DMARC_AddressMode	0x1001_0018h	1	ЧТ/ЗП
DMARC_Control	0x1001_001Ah	4	ЧТ/ЗП
DMARC_InterruptMask	0x1001_001Ch	2	ЧТ/ЗП
DMARC_State	0x1001_001Eh	15	ЧТ/ЗП

8.3.2 Прерывания от контроллера MDMAC

От контроллера MDMAC формируются два запроса на прерывание, которые фиксируются в обоих процессорных системах - NMPU0 и NMPU1:

- Запрос на прерывание по нормальному завершению работы контроллера ПДП выставляется, когда выполняется следующее условие: установлен бит Cpl регистра DMARC_Control и сброшен бит маски MIC в регистре DMARC_InterruptMask (в т. ч. в конце процесса передачи);
- Запрос на прерывание по ошибке доступа в память контроллера ПДП выставляется, когда выполняются одно из условий: установлен бит ES регистра DMATR_Control и сброшен бит маски MIE в регистре DMATR_InterruptMask либо установлен бит ES регистра DMARC_Control и сброшен бит маски MIE в регистре DMARC_InterruptMask.

8.4 Блок разделяемой памяти SHMEM

Блок разделяемой памяти предназначен для организации обмена данными между процессорными системами, а также между процессорными системами и внешними интерфейсами СБИС. Блок представляет собой два банка статической памяти, объемом 256 Кбайт каждый, подключенных к шине AMBA AXI спецификации 3.0. Для каждой из процессорных систем банки разделяемой памяти расположены в карте памяти по адресам:

0 банк - с 0x000A_0000h до 0x000A_FFFFh;
1 банк - с 0x000B_0000h до 0x000B_FFFFh.

Обращение к любому из банков памяти идет независимо от обращения к другому, т. е. обращения к разным банкам могут выполняться одновременно.

Контроллер банка памяти, обеспечивающий подключение банка к шине AMBA AXI, позволяет обращаться к памяти как одиночными, так и пакетными запросами с шириной данных в 8, 16, 32 и 64 бита в соответствии со спецификацией шины.

8.5 Контроллер интерфейса USB

Контроллер USB обеспечивает аппаратную поддержку обмена данными по шине USB и соответствует спецификации устройства типа USB 2.0 High-speed.

Контроллер имеет следующие характеристики:

- Может работать как хост-контроллер (USB host), как конечное устройство (USB device), как OTG устройство;
- Поддержка режимов high-speed (480 Мбит/с) и full-speed (12 Мбит/с).

Особенности в режиме хост-контроллера:

- Поддержка до 16 активных конечных точек;
- Поддержка до 16 host каналов;
- До 8 передач за 1 (микро)фрейм;
- До 127 подключенных устройств;
- Встроенный корневой концентратор (root hub).

Особенности встроенного корневого концентратора:

- Регистр состояния;
- Аппаратная реализация последовательности сброса и определения скорости подключенного устройства.

8.5.1 Внешние выводы контроллера USB

В Таблица 8.3 приведен список внешних выводов микросхемы, относящихся к контроллеру USB.

Таблица 8.3 - Выводы микросхемы, входящие в состав контроллера USB

Вывод	Тип буфера	Примечание
USBCLKSEL	I	Выбор источника тактового сигнала: 0 – тактовый сигнал генерируется с помощью кварцевого осциллятора, 1 – тактовый сигнал генерируется внешним генератором
VBUS	I/O	Питание 5 В USB
ID	I	Идентификатор миниприемника USB
DP	I/O	Линия D+ шины USB
DM	I/O	Линия D– шины USB
TXRTUNE	I/O	Настройка сопротивления передатчика
USB_XI	I/O	Вывод подключения внешнего кварцевого осциллятора 12,0 МГц
USB_XO	I/O	Вывод подключения внешнего кварцевого осциллятора 12,0 МГц или внешнего тактового генератора 12,0 МГц
USBATEST	I/O	Тестирование постоянной составляющей тока
DRVVBUS	O	Управление внешней схемой генерации питания VBUS (OTG режим)

Внешний вывод VBUS позволяет задавать напряжение питания для USB приемников.

Вывод ID определяет, в какой роли (хост-контроллер или конечное устройство) должен выступать данный контроллер при подключении к другому устройству USB и определяется типом коннектора (mini-AB). Если контроллер будет использоваться всегда в одной роли, то значение вывода ID можно оставить неподключенным.

Вывод TXRTUNE предназначен для задания внутреннего сопротивления источника сигнала. Данный вывод должен быть подключен к внешнему резистору номиналом 43,2 Ом ±1 %.

Вывод USBATEST предназначен для тестирования контроллера. При штатной работе данный вывод необходимо оставить неподключенным.

Контроллер тактируется отдельным тактовым сигналом частотой 12 МГц ± 400 ppm.

Если тактовый сигнал блока USB (12 МГц) задается с помощью кварцевого осциллятора, то этот резонатор должен быть подключен к выводам USB_XI и USB_XO.

Если тактовый сигнал блока USB (12 МГц) задается внешним тактовым генератором, то внешний генератор должен быть подключен к выводу USB_XO. Внешний вывод USB_XI при этом должен быть подключен к "земле".

8.5.2 Программно доступные регистры контроллера USB

Программно доступные регистры контроллера USB расположены в области памяти USB DEVICE процессора, и имеют базовое смещение Base = 0x1000_0000h. Спецификация регистров блока представлена в Таблица 8.4.

Таблица 8.4 - Спецификация регистров контроллера USB

Адрес	Начальное значение	Имя	Описание
Общие регистры			
База+0000	02000002h	CONF	Регистр конфигурации контроллера
База+0004	00000000h	MODE	Регистр режима
База+0008	00000000h	INTEN	Регистр маски прерываний
База+000C	00000000h	INTS	Регистр запросов не прерывание
База+0040	00000000h	EPCMD0	Регистры команд конечных точек
База+0044	00000000h	EPCMD1	
База+0048	00000000h	EPCMD2	
База+004C	00000000h	EPCMD3	
База+0050	00000000h	EPCMD4	
База+0054	00000000h	EPCMD5	
База+0058	00000000h	EPCMD6	
База+005C	00000000h	EPCMD7	
База+0060	00000000h	EPCMD8	
База+0064	00000000h	EPCMD9	
База+0068	00000000h	EPCMD10	
База+006C	00000000h	EPCMD11	
База+0070	00000000h	EPCMD12	
База+0074	00000000h	EPCMD13	
База+0078	00000000h	EPCMD14	
База+007C	00000000h	EPCMD15	
Регистры хост-контроллера			
База+0100	00200000h	PORTSC	Регистр управления нисходящим портом
База+0104	00000000h	PORTSTSC	Регистр состояния нисходящего порта
База+0108	00000000h	HOSTEVENTSRC	Регистр состояния хост-контроллера
База+010C	00030000h	HOSTINTEN	Регистр управления прерываниями хост-контроллера
База+0110	00000000h	HCFRMIDX	Регистр счётчика фреймов
База+0114	00000000h	HCFRMINIT	Регистр длительности фрейма
База+0118	00000000h	HCCTRL	Регистр запуска хост-контроллера
База+011C	00000000h	HCSTLINK	Регистр стартового канала

Продолжение таблицы 8.4

Адрес	Начальное значение	Имя	Описание
Регистры конечного устройства			
База+0200	00000020h	DEVC	Регистр управления в режиме конечного устройства
База+0204	00020000h	DEVS	Регистр состояния конечного устройства
База+0208	00000000h	FADDR	Регистр адреса конечного устройства
База+020C	00000000h	TSTAMP	Регистр номера фрейма
Регистры функций OTG			
База+0300	-	OTGC	Регистр управления функциями OTG
База+0310	00000000h	OTGS	Регистр состояния функций OTG
База+0314	00000000h	OTGSTSC	Регистры изменения состояния функций OTG
База+0318	00000000h	OTGSTSFALL	
База+031C	00000000h	OTGSTSRISE	
База+0320	00000000h	OTGTC	Регистр управления таймером OTG
База+0324	00000000h	OTGT	Регистр таймера OTG
Регистры каналов ПДП			
База+0400	00000000h	DMAC1	Регистр управления каналом 1 ПДП
База+0404	00000000h	DMAS1	Регистр состояния кнала 1 ПДП
База+0408	00000000h	DMATCI1	Регистры счётчиков данных канала 1 ПДП
База+040C	00000000h	DMATC1	
База+0420	00000000h	DMAC2	Регистр управления каналом 1 ПДП
База+0424	00000000h	DMAS2	Регистр состояния кнала 1 ПДП
База+0428	00000000h	DMATCI2	Регистры счётчиков данных канала 1 ПДП
База+042C	00000000h	DMATC2	
Управление каналами в режиме хост-контроллера			
База+8000	-	HCEPCTRL1_0	Регистры управления каналами хост-контроллера
База+8008	-	HCEPCTRL1_1	
База+8010	-	HCEPCTRL1_2	
База+8018	-	HCEPCTRL1_3	
База+8020	-	HCEPCTRL1_4	
База+8028	-	HCEPCTRL1_5	
База+8030	-	HCEPCTRL1_6	
База+8038	-	HCEPCTRL1_7	
База+8004	-	HCEPCTRL2_0	
База+800C	-	HCEPCTRL2_1	
База+8014	-	HCEPCTRL2_2	
База+801C	-	HCEPCTRL2_3	
База+8024	-	HCEPCTRL2_4	
База+802C	-	HCEPCTRL2_5	
База+8034	-	HCEPCTRL2_6	
База+803C	-	HCEPCTRL2_7	
База+8040	-	HCEPCONF0	Регистры конфигурации каналов хост-контроллера
База+8044	-	HCEPCONF1	
База+8048	-	HCEPCONF2	
База+804C	-	HCEPCONF3	
База+8050	-	HCEPCONF4	
База+8054	-	HCEPCONF5	
База+8058	-	HCEPCONF6	
База+805C	-	HCEPCONF7	

Продолжение таблицы 8.4

Адрес	Начальное значение	Имя	Описание
База+8080	-	HCEPCOUNT0	Регистры счётчиков каналов хост-контроллера
База+8084	-	HCEPCOUNT1	
База+8088	-	HCEPCOUNT2	
База+808C	-	HCEPCOUNT3	
База+8090	-	HCEPCOUNT4	
База+8094	-	HCEPCOUNT5	
База+8098	-	HCEPCOUNT6	
База+809C	-	HCEPCOUNT7	
База+80A0	-	HCEPCOUNT8	
База+80A4	-	HCEPCOUNT9	
База+80A8	-	HCEPCOUNT10	
База+80AC	-	HCEPCOUNT11	
База+80B0	-	HCEPCOUNT12	
База+80B4	-	HCEPCOUNT13	
База+80B8	-	HCEPCOUNT14	
База+80BC	-	HCEPCOUNT15	
База+80C0	-	HCEPCOUNT16	
База+80C4	-	HCEPCOUNT17	
База+80C8	-	HCEPCOUNT18	
База+80CC	-	HCEPCOUNT19	
База+80D0	-	HCEPCOUNT20	
База+80D4	-	HCEPCOUNT21	
База+80D8	-	HCEPCOUNT22	
База+80DC	-	HCEPCOUNT23	
База+80E0	-	HCEPCOUNT24	
База+80E4	-	HCEPCOUNT25	
База+80E8	-	HCEPCOUNT26	
База+80EC	-	HCEPCOUNT27	
База+80F0	-	HCEPCOUNT28	
База+80F4	-	HCEPCOUNT29	
База+80F8	-	HCEPCOUNT30	
База+80FC	-	HCEPCOUNT31	
Управление конечными точками в режиме конечного устройства			
База+8000	-	EPCTRL0	Регистры управления конечными точками
База+8004	-	EPCTRL1	
База+8008	-	EPCTRL2	
База+800C	-	EPCTRL3	
База+8010	-	EPCTRL4	
База+8014	-	EPCTRL5	
База+8018	-	EPCTRL6	
База+801C	-	EPCTRL7	
База+8020	-	EPCTRL8	
База+8024	-	EPCTRL9	
База+8028	-	EPCTRL10	
База+802C	-	EPCTRL11	
База+8030	-	EPCTRL12	
База+8034	-	EPCTRL13	
База+8038	-	EPCTRL14	
База+803C	-	EPCTRL15	

Продолжение таблицы 8.4

Адрес	Начальное значение	Имя	Описание
База+8040	-	EPCONF0	Регистры конфигурации конечных точек
База+8044	-	EPCONF1	
База+8048	-	EPCONF2	
База+804C	-	EPCONF3	
База+8050	-	EPCONF4	
База+8054	-	EPCONF5	
База+8058	-	EPCONF6	
База+805C	-	EPCONF7	
База+8060	-	EPCONF8	
База+8064	-	EPCONF9	
База+8068	-	EPCONF10	
База+806C	-	EPCONF11	
База+8070	-	EPCONF12	
База+8074	-	EPCONF13	
База+8078	-	EPCONF14	
База+807C	-	EPCONF15	
База+8080	-	EPCOUNT0	Регистры счётчиков конечных точек
База+8084	-	EPCOUNT1	
База+8088	-	EPCOUNT2	
База+808C	-	EPCOUNT3	
База+8090	-	EPCOUNT4	
База+8094	-	EPCOUNT5	
База+8098	-	EPCOUNT6	
База+809C	-	EPCOUNT7	
База+80A0	-	EPCOUNT8	
База+80A4	-	EPCOUNT9	
База+80A8	-	EPCOUNT10	
База+80AC	-	EPCOUNT11	
База+80B0	-	EPCOUNT12	
База+80B4	-	EPCOUNT13	
База+80B8	-	EPCOUNT14	
База+80BC	-	EPCOUNT15	
База+80C0	-	EPCOUNT16	
База+80C4	-	EPCOUNT17	
База+80C8	-	EPCOUNT18	
База+80CC	-	EPCOUNT19	
База+80D0	-	EPCOUNT20	
База+80D4	-	EPCOUNT21	
База+80D8	-	EPCOUNT22	
База+80DC	-	EPCOUNT23	
База+80E0	-	EPCOUNT24	
База+80E4	-	EPCOUNT25	
База+80E8	-	EPCOUNT26	
База+80EC	-	EPCOUNT27	
База+80F0	-	EPCOUNT28	
База+80F4	-	EPCOUNT29	
База+80F8	-	EPCOUNT30	
База+80FC	-	EPCOUNT31	
Память данных			
База+8100 … база+FFFF	-	RAM	Память буферов данных

Регистры со смещением 8000h – 80FCh расположены во встроенном блоке памяти. В разных режимах (хост-контроллер или конечное устройство) спецификация этих регистров отличается. Их значение следует задавать при инициализации контроллера в соответствии с выбранной конфигурацией (режим работы, набор конечных точек и другое). При этом в зарезервированные биты следует явно прописывать значение 0.

По адресам со смещением 8100h – FFFFh расположен банк памяти, используемый для размещения в нём буферов данных конечных точек или каналов хост-контроллера. Формат банка задаётся программно с помощью регистров счётчиков конечных точек и каналов хост-контроллера.

8.5.2.1 Регистр конфигурации контроллера CONF

Формат регистра CONF приведён ниже (см. Таблица 8.5).

Таблица 8.5 - Формат регистра конфигурации контроллера CONF

Биты	Тип	Название поля	Описание
[31:27]	-	-	Зарезервировано
[26]	ЧТ	soft_reset_m	Отражение бита soft_reset
[25]	ЧТ	burst_wait_m	Отражение бита burst_wait
[24]	ЧТ	byte_order_m	Отражение бита byte_order
[23:3]	-	-	Зарезервировано
[2]	ЧТ/ЗП	soft_reset	Программный сброс контроллера USB. Запись значения 1 производит программный сброс контроллера.
[1]	ЧТ/ЗП	burst_wait	Включает дополнительный такт ожидания на шине AHB: 0 – данные идут подряд, 1 – один такт ожидания в операциях на шине AHB.
[0]	ЧТ/ЗП	byte_order	Устанавливает порядок байтов (endianess) в 32-разрядном слове при обращении к контроллеру USB: 0 – младший байт в младшей части слова (little endian), 1 – младший байт в старшей части слова (big endian)

8.5.2.2 Регистр режима MODE

Формат регистра MODE приведён ниже (см. таблицу 8.6).

Биты dev_addr_load_mode и dev_int_mode работают только когда контроллер находится в режиме конечного устройства (dev_en=1)

Таблица 8.6 - Формат регистра MODE

Биты	Тип	Название поля	Описание
[31:4]	-	-	Зарезервировано
[3]	ЧТ/ЗП	dev_addr_load_mode	Используется для установки режима обновления адреса устройства USB: 0 – поле func_addr регистра FADDR необходимо обновлять после завершения статусной фазы запроса SET_ADDRESS, 1 – поле func_addr регистра FADDR обновляется в момент завершения статусной фазы запроса SET_ADDRESS автоматически в то значение, которое было записано заблаговременно.
[2]	ЧТ/ЗП	dev_int_mode	Управляет режимом запросов на прерывание по изменению состояния буферов данных. Для конечной точки 0: 0 – прерывание readyi_int генерируется, когда буфер опустошается после транзакции типа IN, прерывание readyo_int – когда первое данное поступает в буфер во время транзакции типа OUT 1 – прерывание readyi_int или readyo_int генерируется, когда происходит запись в бит bufwr или bufrd регистра EPCMD (в том случае, когда соответствующий буфер доступен для записи или чтения), прерывание readyi_int также генерируется в момент записи бита init. Для конечных точек 1-15: 0 – прерывание ready_int генерируется, когда буфер перестаёт быть полным (при транзакции типа IN), прерывание empty_int генерируется, когда буфер становится пустым (при транзакции типа IN), прерывание ready_int генерируется, когда первое данное поступает в буфер (при транзакции типа OUT), 1 – прерывание ready_int генерируется, когда происходит запись в бит bufwr или bufrd регистра EPCMD (в том случае, когда соответствующий буфер доступен для записи или чтения), в этом случае также запись в бит init производит генерацию прерываний empty_int и ready_int для транзакций типа IN и прерывания empty_int для транзакций типа OUT.
[1]	ЧТ/ЗП	dev_en	Запись значения 1 в бит dev_en переводит контроллер в режим конечного устройства.
[0]	ЧТ/ЗП	host_en	Запись значения 1 в бит host_en переводит контроллер в режим хост-контроллера.

8.5.2.3 Регистр маски прерываний INTEN

Формат регистра INTEN приведён ниже (см. Таблица 8.7).

Таблица 8.7 - Формат регистра INTEN

Биты	Тип	Название поля	Описание
[31:16]	ЧТ/ЗП	dev_ep_inten	Разрешение прерывания dev_ep_int
[15:10]	-	Зарезервировано	
[9]	ЧТ/ЗП	dma2_inten	Разрешение прерывания dma2_int
[8]	ЧТ/ЗП	dma1_inten	Разрешение прерывания dma1_int
[7:5]	-	-	Зарезервировано
[4]	ЧТ/ЗП	cmd_inten	Разрешение прерывания cmd_int
[3]	ЧТ/ЗП	phy_err_inten	Разрешение прерывания phy_err_int
[2]	ЧТ/ЗП	otg_inten	Разрешение прерывания otg_int
[1]	ЧТ/ЗП	dev_inten	Разрешение прерывания dev_int
[0]	ЧТ/ЗП	host_inten	Разрешение прерывания host_int

Каждый бит регистра INTEN включает прерывание по установке соответствующего бита регистра INTS. Если разряд регистра INTEN имеет значение 1, то прерывание по установке соответствующего бита регистра INTS разрешено. Если разряд регистра INTEN имеет значение 0, то прерывание по установке соответствующего бита регистра INTS запрещено.

8.5.2.4 Регистр запросов на прерывание INTS

Формат регистра INTS приведён ниже (см. Таблица 8.8).

Таблица 8.8 - Формат регистра INTS

Биты	Тип	Название поля	Описание
[31:16]	ЧТ	dev_ep_int	Биты поля dev_ep_int инициируют запрос на прерывание от конечных точек контроллера. Причиной установки бита 16 (конечная точка 0) служит одно из следующих условий: - готовность буфера для транзакции типа IN, - готовность буфера для транзакции типа OUT, - получение пакета PING, - выдача пакета NAK, - выдача пакета STALL.
			Причиной установки каждого из битов 17-31 (конечные точки 1-15 соответственно) служит одно из следующих условий: - готовность буфера для транзакции типа IN, - готовность буфера для транзакции типа OUT, - получение пакета PING, - выдача пакета NAK, - выдача пакета STALL. - буфер становится пустым. Поле dev_ep_int сбрасывается также при сбросе шины USB.

Продолжение таблицы 8.8

Биты	Тип	Название поля	Описание
[15:10]	-	-	Зарезервировано
[9]	ЧТ/ЗП	dma2_int	Биты dma1_int и dma2_int инициируют запросы на прерывание от каналов ПДП (1 и 2 соответственно). Причиной установки бита dma1_int или dma2_int служит одно из следующих условий: - канал ПДП завершил работу, - канал ПДП завершил работу с ошибкой, - канал ПДП был остановлен.
[8]	ЧТ/ЗП	dma1_int	
[7:5]	-	-	Зарезервировано
[4]	ЧТ/ЗП	cmd_int	Бит cmd_int инициируют запрос на прерывание при завершении записи в регистр EPCMD.
[3]	ЧТ/ЗП	phy_err_int	Бит phy_err_int сигнализирует об аппаратном сбое блока физического кодирования/декодирования.
[2]	ЧТ	otg_int	Бит otg_int сигнализирует об одном из следующих событий: - состояние SE0 (низкий уровень на обеих сигнальных линиях USB) продолжительностью более 2 мс, - изменение уровня на входе USBID, - таймер OTG (регистр OTGT) досчитал до 0.
[1]	ЧТ	dev_int	Бит dev_int устанавливается при наступлении одного из следующих событий: - переход или выход из состояния пониженного энергопотребления (Suspend), - приём пакета SOF, - приём пакета SETUP, - сброс шины USB.
[0]	ЧТ	host_int	Бит host_en устанавливается при наступлении одного из следующих событий: - передача пакета SOF, - переполнение счётчика фреймов, - изменение состояния подключения, - ошибка подключения, - переход или выход из состояния пониженного энергопотребления (Suspend), - завершение сброса шины USB, - завершена операция со всеми конечными точками.

Биты данного регистра устанавливаются аппаратно, что может вызывать запрос на прерывание к процессорной системе. Сброс битов, доступных только для чтения (dev_ep_int, otg_int, dev_int, host_int), производится также аппаратно при устранении причины установки. Сброс битов, достпных для записи (dma2_int, dma1_int, cmd_int, phy_err_int), производится программно путём записи в даный бит значения 0.

8.5.2.5 Регистры команд конечных точек EPCMD0-EPCMD15

Имеется 16 регистров EPCMD для каждой конечной точки или канала хост-контроллера.

Регистр EPCMD имеет разный формат в разных режимах контроллера. Формат регистра EPCMD для режима хост-контроллера (бит host_en регистра MODE установлен) приведён ниже (см. Таблица 8.9 и Таблица 8.10).

Таблица 8.9 - Формат регистра EPCMD для режима хост-контроллера

Биты	Тип	Название поля	Описание
[31]	ЧТ	epcmd_busy	Поле epcmd_busy показывает готовность регистра принять следующую запись: 0 – запись в регистр EPCMD разрешена, 1 – регистр занят. Для записи в регистр EPCMD необходимо дождаться, пока бит epcmd_busy станет равным 0, и только затем производить запись. Есть также 2 способа записи в данный регистр, в зависимости от записываемого значения бита write_en: при write_en=1 обновляются все поля регистра EPCMD, при write_en=0 обновляются только биты 0-8 и 18-22.
[30]	-	-	Зарезервировано
[29:28]	ЗП	speed	Поле speed задаёт скорость канала: 00 – full speed, 01 – low speed, 10 – high speed, 11 – запрещенный код.
[27:26]	ЗП	bnum	Поле bnum задаёт тип буферизации: 00 – одинарная, 01 – двойная, 10, 11 – запрещенный код.
[25]	ЗП	sc	Поле sc задаёт тип пакета SPLIT (если используется): 0 – Start-Split, 1 – Complete-Split.
[24:23]	ЗП	et	Поле et задаёт тип транзакции USB: 00 – Control, 01 – Isochronous, 10 – Bulk, 11 – Interrupt.
[22:20]	-	-	Зарезервировано
[19]	ЗП	status_clr	Запись значения 1 в бит status_clr сбрасывает поле status в регистре HCEPCTRL1
[18]	ЗП	errcnt_clr	Запись значения 1 в бит errcnt_clr сбрасывает поле errcnt в регистре HCEPCTRL1.
[17:16]	ЗП	sendpid	Поле sendpid устанавливает тип передачи: 00 – OUT, 01 – IN, 10 – SETUP, 11 – PING.
[15]	ЗП	nlinkinvalid	Бит nlinkinvalid устанавливает значение бита nlinkinvalid регистра HCEPCTRL1.

Продолжение таблицы 8.9

Биты	Тип	Название поля	Описание
[14:12]	ЗП	nextlink	Поле nextlink устанавливает значение поля nextlink регистра HCEPCTRL1.
[11]	ЗП	write_en	0 - разрешена запись полей регистра EPCMD[22:18] и EPCMD[8:0] 1 - разрешена запись полей регистра EPCMD[29:0]
[10:9]	-	-	Зарезервировано
[8]	ЗП	toggle_clr	Запись значения 1 в бит toggle_clr сбрасывает секвенсор пакетов данных в канале (бит toggle регистра HCEPCTRL1 в значение 0).
[7]	ЗП	toggle_set	Запись значения 1 в бит toggle_set устанавливает секвенсор пакетов данных в канале (бит toggle регистра HCEPCTRL1 в значение 1).
[6:5]	-	-	Зарезервировано
[4]	ЗП	bufrd	Запись значения 1 в бит bufrd даёт сигнал о завершении чтения из буфера канала.
[3]	ЗП	bufwr	Запись значения 1 в бит bufwr даёт сигнал о завершении записи в буфер канала.
[2]	ЗП	init	Запись значения 1 в бит init инициализирует буфер канала.
[1]	ЗП	stop	Запись значения 1 в бит stop останавливает канал (устанавливает trans_en=0 в регистре HCEPCTRL1).
[0]	ЗП	start	Запись значения 1 в бит start запускает канал (устанавливает trans_en=1 в регистре HCEPCTRL1).

Таблица 8.10 - Формат регистра EPCMD для режима конечного устройства

Биты	Тип	Название поля	Описание
[31]	ЧТ	epcmd_busy	Поле epcmd_busy показывает готовность регистра принять запись: 0 – запись в регистр EPCMD разрешена, 1 – регистр занят. Для записи в регистр EPCMD необходимо дождаться, пока бит epcmd_busy станет равным 0, и только затем производить запись. Есть также 2 способа записи в данный регистр, в зависимости от записываемого значения бита write_en: при write_en=1 обновляются все поля регистра EPCMD, при write_en=0 обновляются только биты 0-8 и 18-22.
[30]	-	-	Зарезервировано
[29:28]	ЗП	hiband	Поле hiband задаёт количество пакетов передаваемое или получаемое за один (микро)фрейм: 00 – 1, 01 – 1, 10 – 2, 11 – 3.
[27:26]	ЗП	bnum	Поле bnum задаёт тип буферизации: 00 – одинарная, 01 – двойная, 10, 11 – запрещенный код.
[25]	ЗП	dir	Поле dir задаёт направление транзакции: 0 – OUT, 1 – IN.
[24:23]	ЗП	et	Поле et задаёт тип транзакции: 00 – запрещенный код, 01 – Isochronous, 10 – Bulk, 11 – Interrupt.

Продолжение таблицы 8.10

Биты	Тип	Название поля	Описание
[22]	ЗП	nack_int_clr	Запись значения 1 в бит nack_int_clr снимает бит nack_int регистра EPCTRL (снимает причину прерывания по отправке пакета NAK).
[21]	ЗП	stalled_int_clr	Запись значения 1 в бит stalled_int_clr снимает бит stalled_int регистра EPCTRL (снимает причину прерывания по отправке пакета STALL).
[20]	ЗП	ping_int_clr	Запись значения 1 в бит ping_int_clr снимает бит ping_int регистра EPCTRL (снимает причину прерывания по приёму пакета PING).
[19]	ЗП	readyo_int_clr /empty_int_clr	Запись значения 1 в бит readyo_int_clr/empty_int_clr снимает бит readyo_int/empty_int регистра EPCTRL0/EPCTRL1-EPCTRL15 (снимает причину прерывания по готовности буфера конечной точки 0/1-15).
[18]	ЗП	readyi_int_clr/ ready_int_clr	Запись значения 1 в бит readyi_int_clr/ready_int_clr снимает бит readyi_int/ready_int регистра EPCTRL0/EPCTRL1-EPCTRL15 (снимает причину прерывания по готовности буфера конечной точки 0/1-15).
[17]	-	-	Зарезервировано
[16]	ЗП	nack_inten	Бит nack_inten включает прерывание по передаче пакета NAK: 1 – включено, 0 – выключено.
[15]	ЗП	stalled_inten	Бит stalled_inten включает прерывание по передаче пакета STALL: 1 – включено, 0 – выключено.
[14]	ЗП	ping_inten	Бит ping_inten включает прерывание по приёму пакета PING: 1 – включено, 0 – выключено.
[13]	ЗП	readyo_int_en /empty_inten	Бит readyo_int_en/empty_inten включает прерывание по установке бита readyo_int/empty_int регистра EPCTRL0/EPCTRL1-EPCTRL15.
[12]	ЗП	readyi_int_en/ ready_inten	Бит readyi_int_en/ready_int_en включает прерывание по установке бита readyi_int/ready_int регистра EPCTRL0/EPCTRL1-EPCTRL15.
[11]	ЗП	write_en	
[10]	ЗП	nackresp	Запись значения 1 в бит nackresp инициирует выдачу ответа NAK на транзакцию Bulk IN и OUT.
[9]	ЗП	nullresp	Запись значения 1 в бит nullresp инициирует выдачу пустого пакета в ответ на транзакцию Bulk или Interrupt IN.
[8]	ЗП	toggle_clr	Запись значения 1 в бит toggle_clr сбрасывает секвенсор пакетов данных конечной точки (бит toggle регистра EPCTRL в значение 0).
[7]	ЗП	toggle_set	Запись значения 1 в бит toggle_set устанавливает секвенсор пакетов данных конечной точки (бит toggle регистра EPCTRL в значение 1).
[6]	ЗП	stall_clr	Запись значения 1 в бит stall_clr сбрасывает бит stall регистра EPCTRL.
[5]	ЗП	stall_set	Запись значения 1 в бит stall_set устанавливает сбит stall регистра EPCTRL.
[4]	ЗП	bufrd	Запись значения 1 в бит bufrd даёт сигнал о завершении чтения из буфера конечной точки.
[3]	ЗП	bufwr	Запись значения 1 в бит bufwr даёт сигнал о завершении записи в буфер конечной точки.
[2]	ЗП	init	Запись значения 1 в бит init инициализирует буфер конечной точки.
[1]	ЗП	stop	Запись значения 1 в бит stop останавливает конечную точку (устанавливает ep_en=0 в регистре EPCTRL).
[0]	ЗП	start	Запись значения 1 в бит start запускает конечную точку (устанавливает ep_en=1 в регистре EPCTRL).

8.5.2.6 Регистр управления нисходящим портом PORTSC

Формат регистра PORTSC приведён ниже (см. Таблица 8.11).

Таблица 8.11 - Формат регистра PORTSC

Биты	Тип	Название поля	Описание
[31]	ЧТ/ЗП	port_resume_req	Запись значения 1 в бит port_resume_req инициирует возвращение порта в активное состояние (Resume).
[30]	ЧТ/ЗП	port_suspend_req	Запись значения 1 в бит port_suspend_req инициирует переход порта в состояние пониженного энергопотребления (Suspend). Переход осуществляется в конце текущего фрейма.
[29]	ЧТ/ЗП	port_wakeup_req	Бит port_wakeup_req включает возможность внешнего устройства инициировать выход из режима пониженного энергопотребления (функция Remote Wakeup): 0 – выключено, 1 – включено (Remote Wakeup используется).
[28:27]	-	-	Зарезервировано
[26]	ЧТ/ЗП	port_enable_req	Запись значения 1 в бит port_enable_req включает порт.
[25]	ЧТ/ЗП	port_reset_req	Запись значения 1 в бит port_reset_req инициирует сброс порта.
[24]	-	-	Зарезервировано
[23]	ЧТ/ЗП	forcefs_req	Бит forcefs_req отключает механизм автоопределения high-speed устройств (chirp-последовательность): 0 – устройство пытается подключиться как high-speed, 1 – устройство остаётся full-speed.
[22]	ЧТ/ЗП	port_power_req	Бит port_power_req управляет подачей питания на шину Vbus порта: 0 – питание отключено, 1 – питание включено.
[21]	ЧТ/ЗП	port_power_ctl_req	Бит port_power_ctl_req устанавливает способ отключения питания при повышенном потреблении: 1 – с помощью данного контроллера, 0 – с помощью внешнего предохранителя
[20:12]	-	-	Зарезервировано
[11]	ЧТ	port_resuming_rhs	Бит port_resuming_rhs имеет значение 1 в момент возвращения из режима пониженного энергопотребления, значение 0 – в остальное время.

Продолжение таблицы 8.11

Биты	Тип	Название поля	Описание
[10]	ЧТ	port_suspended_rhs	Бит port_suspended_rhs показывает, находится ли контроллер в режиме пониженного энергопотребления: 1 – находится (Sespend), 0 – в остальное время.
[9]	ЧТ	port_wakeup_rhs	Бит port_wakeup_rhs отображает значение бита port_wakeup_req.
[8]	ЧТ	port_high_speed_rhs	Бит port_high_speed_rhs показывает, подключено ли к порту устройство типа high-speed: 1 – подключенное устройство прошло процесс автоопределения скорости high-speed, 0 – устройство не прошло процесс автоопределения скорости high-speed.
[7]	ЧТ	port_low_speed_rhs	Бит port_low_speed_rhs показывает, подключено ли к порту устройство типа low-speed: 1 – к порту подключено устройство типа low-speed, 0 – в других случаях.
[6]	ЧТ	port_enable_rhs	Бит port_enable_rhs показывает, находится ли порт во включенном состоянии: 1 – порт включен и прошёл процедуру сброса, 0 – в других случаях.
[5]	ЧТ	port_reset_rhs	Бит port_reset_rhs показывает, находится ли порт в состоянии сброса: 1 – порт сбрасывается в данный момент, 0 – в других случаях.
[4]	ЧТ	port_connection_rhs	Бит port_connection_rhs показывает наличие подключения к порту: 1 – устройство (независимо от типа) подключено к порту, 0 – нет подключенного устройства.
[3]	ЧТ	forcefs_rhs	Бит forcefs_rhs отображает значение бита forcefs_req.
[2]	ЧТ	port_power_rhs	Бит port_power_rhs отображает значение бита port_power_req.
[1]	ЧТ	port_power_ctl_rhs	Бит port_power_ctl_rhs отображает значение бита port_power_ctl_req.
[0]		-	Зарезервировано

8.5.2.7 Регистр состояния нисходящего порта PORTSTSC

Формат регистра PORTSTSC приведён ниже (см. Таблица 8.12).

Таблица 8.12 - Формат регистра PORTSTSC

Биты	Тип	Название поля	Описание
[31:7]	-		Зарезервировано
[6:5]	ЧТ	linestate	Поле linestate показывает текущее состояние выводов D+/D- (DPLUS – бит 5, DMINUS – бит 6).
[4]	ЧТ/ЗП	port_reset_c	Бит port_reset_c устанавливается в момент завершения сброса порта.
[3]	ЧТ/ЗП	port_ov_curr_c	Бит port_ov_curr_c устанавливается в момент изменения состояния вывода OVCUR.
[2]	ЧТ/ЗП	port_suspend_c	Бит port_suspend_c устанавливается в момент выхода из состояния пониженного энергопотребления (Suspend).
[1]	ЧТ/ЗП	port_enable_c	Бит port_enable_c устанавливается при выключении порта из-за ошибки подключения.
[0]	ЧТ/ЗП	port_connection_c	Бит port_connection_c устанавливается при подключении и отключении порта.

Биты регистра PORTSTSC[4:0] устанавливаются аппаратно и сбрасываются программно путём записи в них значения 0.

8.5.2.8 Регистр состояния хост-контроллера HOSTEVENTSRC

Формат регистра HOSTEVENTSRC приведён ниже (см. Таблица 8.13).

Таблица 8.13 - Формат регистра HOSTEVENTSRC

Биты	Тип	Название поля	Описание
[31:16]	-	-	Зарезервировано
[15:8]	ЧТ/ЗП	trans_done	Поле trans_done показывает завершение работы каналов хост-контроллера. Бит поля устанавливается автоматически при завершении передачи соответствующим каналом хост-контроллера (когда сбрасывается бит trans_en регистра HCEPCTRL1). Сброс битов trans_done производится программно путём записи в них значения 0.
[2]	ЧТ	dport_evt	Бит dport_evt объединяет по ИЛИ биты port_reset_c, port_ov_curr_c, port_suspend_c, port_enable_c, port_connection_c.
[1]	ЧТ/ЗП	frameov	Бит frameov устанавливается в момент переполнения счётчика фреймов, сброс бита производится программно путём записи в него значения 0.
[0]	ЧТ/ЗП	sofstart	Бит sofstart устанавливается автоматически через заданное число микрофреймов (в поле sofevtinterval регистра HOSTINTEN).

8.5.2.9 Регистр управления прерываниями хост-контроллера HOSTINTEN

Формат регистра HOSTINTEN приведён ниже (см. Таблица 8.14).

Таблица 8.14 - Формат регистра HOSTINTEN

Биты	Тип	Название поля	Описание
[31:19]	-	-	Зарезервировано
[18:16]	ЧТ/ЗП	sofevtinterval	Поле sofevtinterval задаёт период, с которым устанавливается значение 1 в бите sofstart регистра HOSTEVENTSRC: 000 – 1 микрофрейм (125 мкс), 001 – 2 микрофрейма (250 мкс), 010 – 4 микрофрейма (500 мкс), 011 – 8 микрофреймов (1 мс, 1 фрейм), 100 – 16 микрофреймов (2 мс), 101 – 32 микрофрейма (4 мс), 110 – 64 микрофрейма (8 мс), 111 – 128 микрофреймов (16 мс).
[15:8]	ЧТ/ЗП	trans_done_inten	Разрешение прерывания по установке бита trans_done
[7]	-	-	Зарезервировано
[6]	ЧТ/ЗП	port_reset_c_inten	Разрешение прерывания по установке бита port_reset_c
[5]	ЧТ/ЗП	port_ov_curr_c_inten	Разрешение прерывания по установке бита port_ov_curr_c
[4]	ЧТ/ЗП	port_suspend_c_inten	Разрешение прерывания по установке бита port_suspend_c
[3]	ЧТ/ЗП	port_enable_c_inten	Разрешение прерывания по установке бита port_enable_c
[2]	ЧТ/ЗП	port_connection_c_inten	Разрешение прерывания по установке бита port_connection_c
[1]	ЧТ/ЗП	frameov_inten	Разрешение прерывания по установке бита frameov
[0]	ЧТ/ЗП	sofstart_inten	Разрешение прерывания по установке бита sofstart

Биты 0-15 регистра HOSTINTEN включают прерывание по установке соответствующих битов регистра HOSTEVENTSRC и PORTSTSC.

Если разряд регистра HOSTINTEN имеет значение 1, то прерывание по установке соответствующего бита регистра HOSTEVENTSRC или PORTSTSC разрешено.

Если разряд регистра HOSTINTEN имеет значение 0, то прерывание по установке соответствующего бита регистра HOSTEVENTSRC или PORTSTSC запрещено.

8.5.2.10 Регистр счётчика фреймов HCFRMIDX

Формат регистра HCFRMIDX приведён ниже (см. Таблица 8.15).

Таблица 8.15 - Формат регистра HCFRMIDX

Биты	Тип	Название поля	Описание
[31:27]	-	-	Зарезервировано
[26:16]	ЧТ	sofv	Поле sofv – старшие 11 разрядов счётчика frmidx и используется как номер фрейма в пакетах SOF
[15:14]	-	-	Зарезервировано
[13:0]	ЧТ/ЗП	frmidx	Поле frmidx считает микрофреймы – инкрементируется каждые 125 мкс.

8.5.2.11 Регистр длительности фрейма HCFRMINIT

Формат регистра HCFRMINIT приведён ниже (см. Таблица 8.16).

Таблица 8.16 - Формат регистра HCFRMINIT

Биты	Тип	Название поля	Описание
[31:16]	ЧТ	frmcnt	Поле frmcnt показывает текущее значение обратного счётчика, отсчитывающего интервал микрофрейма.
[15:13]	-	-	Зарезервировано
[12:0]	ЧТ/ЗП	frminit	Поле frminit задаёт интервал микрофрейма в периодах тактового сигнала 60 МГц. Значение данного поля после сброса – 7499 (1D4Bh) – соответствует стандортному значению 125 мкс. Изменять значение данного поля может понадобиться только в целях отладки.

8.5.2.12 Регистр запуска хост-контроллера HCCTRL

Формат регистра HCCTRL приведён ниже (см. Таблица 8.17).

Таблица 8.17 - Формат регистра HCCTRL

Биты	Тип	Название поля	Описание
[31:1]	-	-	Зарезервировано
[0]	ЧТ/ЗП	hcrun	Бит hcrun производит запуск хост-контроллера: 1 – контроллер запущен, 0 – контроллер приостановлен.

8.5.2.13 Регистр стартового канала HCSTLINK

Формат регистра HCSTLINK приведён ниже (см. Таблица 8.18).

Таблица 8.18 - Формат регистра HCSTLINK

Биты	Тип	Название поля	Описание
[31:3]	-	-	Зарезервировано
[2:0]	ЧТ/ЗП	startlink	Поле startlink задаёт номер канала (номер регистра HCEPCTRL1), с которого хост-контроллер начинает работу.

8.5.2.14 Регистр управления в режиме конечного устройства DEVC

Формат регистра DEVC приведён ниже (см. Таблица 8.19).

Таблица 8.19 - Формат регистра DEVC

Биты	Тип	Название поля	Описание
[31]	ЧТ/ЗП	status_ng_inten	Биты *_inten включают генерацию запросов на прерывание по установке соответствующих битов *_int регистра DEVS.
[30]	ЧТ/ЗП	status_ok_inten	
[29]	ЧТ/ЗП	usbrstb_inten	
[28]	ЧТ/ЗП	usbrste_inten	
[27]	ЧТ/ЗП	setup_inten	
[26]	ЧТ/ЗП	sof_inten	
[25]	ЧТ/ЗП	suspendb_inten	
[24]	ЧТ/ЗП	suspende_inten	
[23:17]	-	-	Зарезервировано
[16]	ЧТ/ЗП	physusp	Бит physusp переводит блок физического интерфейса в состояние пониженного энергопотребления.
[15:6]	-	-	Зарезервировано
[5]	ЧТ/ЗП	disconnect	Бит disconnect производит отключение устройства от порта USB (электрическим способом, снимая воздействия на порт концентратора).
[4]	-	-	Зарезервировано
[3]	ЧТ/ЗП	enrmtwkup	Бит enrmtwkup разрешает режим Remote Wakeup.
[2]	ЧТ/ЗП	reqresume	Запись значения 1 в бит reqresume инициирует запрос Remote Wakeup.
[1]	-	-	Зарезервировано
[0]	ЧТ/ЗП	reqspeed	Бит reqspeed устанавливает режим физического интерфейса: 0 – high-speed, 1 – full-speed.

8.5.2.15 Регистр состояния конечного устройства DEVS

Формат регистра DEVS приведён ниже (см. Таблица 8.20).

Таблица 8.20 - Формат регистра DEVS

Биты	Тип	Название поля	Описание
[31]	ЧТ/ЗП	status_ng_int	Бит status_ng_int устанавливается в момент ошибки управляющих транзакций USB. Отслеживаются следующие ситуации: выдача пакета STALL в ходе фазы данных или статуса (Data Stage, Status Stage), получение пакета SETUP до завершения предыдущей фазы статуса, непустой пакет в фазе статуса, таймаут, неверное направление передач или неверный секвенсор данных в управляющей транзакции.
[30]	ЧТ/ЗП	status_ok_int	Бит status_ok_int устанавливается в момент корректного завершения фазы статуса управляющей транзакции USB.
[29]	ЧТ/ЗП	usbrstb_int	Бит usbrstb_int устанавливается в начале процедуры сброса USB.
[28]	ЧТ/ЗП	usbrste_int	Бит usbrste_int устанавливается в конце процедуры сброса USB.
[27]	ЧТ/ЗП	setup_int	Бит setup_int устанавливается в момент начала управляющей фазы (Setup Stage) управляющей транзакции USB.
[26]	ЧТ/ЗП	sof_int	Бит sof_int устанавливается в момент приёма пакета SOF или, если контроллер не принял пакет SOF, когда контроллер определяет начало (микро)фрейма с помощью собственного механизма.
[25]	ЧТ/ЗП	suspendb_int	Бит suspendb_int устанавливается в момент перехода в состояние пониженного энергопотребления.
[24]	ЧТ/ЗП	suspende_int	Бит suspende_int устанавливается в момент выхода из состояния пониженного энергопотребления.
[23:18]	-	-	Зарезервировано
[17]	ЧТ	crtspeed	Бит crtspeed показывает текущую настройку скорости: значение 0 – high-speed, 1 – full-speed.
[16]	ЧТ	phyreset	Бит phyreset установлен во время сброса блока физического интерфейса.
[15:2]	-	-	Зарезервировано
[1]	ЧТ	busreset	Бит busreset установлен во время сброса USB.
[0]	ЧТ	suspend	Бит suspend установлен на время режима пониженного энергопотребления.

Сброс битов *_int производится программно путём записи в них значения 0.

8.5.2.16 Регистр адреса конечного устройства FADDR

Формат регистра FADDR приведён ниже (см. Таблица 8.21).

Таблица 8.21 - Формат регистра FADDR

Биты	Тип	Название поля	Описание
[31:23]	-	-	Зарезервировано
[22:16]	ЧТ	crt_func_addr	Поле crt_func_addr показывает текущий установленный адрес устройства на шине USB.
[15:9]	-	-	Зарезервировано
[8]	ЧТ/ЗП	dev_configured	Бит dev_configured переводит устройство в состояние "сконфигурировано" (Configured). Это бит следует устанавливать после обработки запроса SET_CONFIGURATION.
[7]	-	-	Зарезервировано
[6:0]	ЧТ/ЗП	func_addr	Поле func_addr следует использовать для установки адреса USB устройства после обработки запроса SET_ADDRESS.

8.5.2.17 Регистр номера фрейма TSTAMP

Формат регистра TSTAMP приведён ниже (см. Таблица 8.22).

Таблица 8.22 - Формат регистра TSTAMP

Биты	Тип	Название поля	Описание
[31:11]	-	-	Зарезервировано
[10:0]	ЧТ	timestamp	Поле timestamp показывает номер текущего фрейма (значение, принятое от хост-контроллера, или полученное с помощью собственного механизма).

8.5.2.18 Регистр управления функциями OTG OTGC

Формат регистра OTGC приведён ниже (см. Таблица 8.23).

Таблица 8.23 - Формат регистра OTGC

Биты	Тип	Название поля	Описание
[31:10]	-	-	Зарезервировано
[9]	ЧТ/ЗП	id_pull_up	Бит id_pull_up управляет включением подтягивающего резистора вывода ID: 0 – подтягивающий резистор отключен от питания (ключ разомкнут), 1 – подтягивающий резистор подключен к питанию (ключ замкнут).
[8]	ЧТ/ЗП	dp_pull_down	Бит dp_pull_down управляет включением подтягивающего резистора вывода USBVP: 0 – подтягивающий резистор отключен от земли (ключ разомкнут), 1 – подтягивающий резистор подключен к земле (ключ замкнут).
[7]	ЧТ/ЗП	dm_pull_down	Бит dm_pull_down управляет включением подтягивающего резистора вывода USBVM: 0 – подтягивающий резистор отключен от земли (ключ разомкнут), значение 1 – подтягивающий резистор подключен к земле (ключ замкнут).
[6:0]	-	-	Зарезервировано

8.5.2.19 Регистр состояния функций OTG OTGS

Формат регистра OTGS приведён ниже (см. Таблица 8.24).

Таблица 8.24 - Формат регистра OTGS

Биты	Тип	Название поля	Описание
[31:11]	-	-	Зарезервировано
[10]	ЧТ	vbus_vld	Бит vbus_vld показывает подключение устройства к шине питания USB, значение соответствует выводу USBCON.
[9:7]	-	-	Зарезервировано
[6]	ЧТ	id	Бит id показывает значение на выводе USBID: 0 – вывод USBID подключен к земле (подключен разъём mini-A), 1 – в остальных случаях (подтянут к питанию с помощью резистора, управляемого битом id_pull_up).
[5:1]	-	-	Зарезервировано
[0]	ЧТ	otg_tmrout	Бит otg_tmrout устанавливается в момент, когда таймер OTG (управляемый регистрами OTGT и OTGTC) досчитал до нуля.

8.5.2.20 Регистры изменения состояния функций OTG OTGSTSC, OTGSTSFALL, OTGSTSRISE

Формат регистра OTGSTSC приведён ниже (см. Таблица 8.25).

Таблица 8.25 - Формат регистра OTGSTSC

Биты	Тип	Название поля	Описание
[31:11]	-	-	Зарезервировано
[10]	ЧТ/ЗП	vbus_vld_c	Бит vbus_vld устанавливается в следующих случаях: в момент перехода уровня USBCON из 0 в 1, если установлен бит vbus_vld_ren, в момент перехода уровня USBCON из 1 в 0, если установлен бит vbus_vld_fen.
[9:7]	-	-	Зарезервировано
[6]	ЧТ/ЗП	id_c	Бит id_c устанавливается в следующих случаях: в момент перехода уровня USBID из 0 в 1, если установлен бит id_ren, в момент перехода уровня USBID из 1 в 0, если установлен бит id_fen.
[5:1]	-	-	Зарезервировано
[0]	ЧТ/ЗП	otg_tmrout_c	Бит otg_tmrout_c устанавливается в следующем случае: в момент перехода бита otg_tmrout регистра OTGS из 0 в 1, если установлен бит otg_tmrout_ren,

Биты регистра OTGSTSC устанавливаются автоматически и сбрасываются программно путём записи в них значения 0.

Форматы регистров OTGSTSFALL и OTGSTSRISE приведены ниже (см. Таблица 8.26 и Таблица 8.27).

Таблица 8.26 - Формат регистра OTGSTSFALL

Биты	Тип	Название поля	Описание
[31:11]	-	-	Зарезервировано
[10]	ЧТ/ЗП	vbus_vld_fen	Разрешение установки vbus_vld_c при изменении vbus_vld 0->1
[9:7]	-	-	Зарезервировано
[6]	ЧТ/ЗП	id_fen	Разрешение установки id_c при изменении id 0->1
[5:0]	-	-	Зарезервировано

Таблица 8.27 - Формат регистра OTGSTSRISE

Биты	Тип	Название поля	Описание
[31:11]	-	-	Зарезервировано
[10]	ЧТ/ЗП	vbus_vld_ren	Разрешение установки vbus_vld_c при изменении vbus_vld 1->0
[9:7]	-	-	Зарезервировано
[6]	ЧТ/ЗП	id_ren	Разрешение установки id_c при изменении id 1->0
[5:1]	-	-	Зарезервировано
[0]	ЧТ/ЗП	otg_tmrout_ren	

8.5.2.21 Регистр управления таймером OTG OTGTC

Формат регистра OTGTC приведён ниже (см. Таблица 8.28).

Таблица 8.28 - Формат регистра OTGTC

Биты	Тип	Название поля	Описание
[31:16]	-	-	Зарезервировано
[15:8]	ЧТ/ЗП	tmr_prescale	Поле tmr_prescale задаёт делитель частоты тактвого сигнала 60 МГц для формирования единицы отсчёта в 0,01 мс для таймера OTG. Значение после сброса – 149. Изменять данное значение может понадобиться только для отладки.
[7:2]	-	-	Зарезервировано
[1]	ЧТ/ЗП	tmr_mode	Бит tmr_mode устанавливает режим работы таймера OTG: 0 – однократный запуск, 1 – постоянная работа.
[0]	ЧТ/ЗП	start_tmr	Бит start_tmr производит запуск таймера: 1 – таймер запущен, 0 – таймер остановлен.

8.5.2.22 Регистр таймера OTG OTGT

Формат регистра OTGT приведён ниже (см. Таблица 8.29).

Таблица 8.29 - Формат регистра OTGT

Биты	Тип	Название поля	Описание
[31:16]	-	-	Зарезервировано
[15:0]	ЧТ/ЗП	tmr_init_val	Поле tmr_init_val задаёт период счёта таймера OTG в единицах отсчёта, заданных в поле tmr_prescale регистра OTGTC.

8.5.2.23 Регистры управления ПДП DMAC1, DMAC2

Контроллер содержит два канала ПДП: 1 и 2. Для каждого имеются регистры одинакового формата, представленного ниже (см. Таблица 8.30). Под символом X понимается 1 или 2 для соответствующего канала.

Таблица 8.30 - Формат регистров DMACX

Биты	Тип	Название поля	Описание
[31:27]	-	-	Зарезервировано
[26:16]	ЧТ/ЗП	dmaX_blksize	Поле dmaX_blksize настраивает размер передачи в байтах за один запрос контроллеру ПДП.
[15:12]	-	-	Зарезервировано
[11:8]	ЧТ/ЗП	dmaX_ep	Поле dmaX_ep устанавливает номер конечной точки, с которой будет работать данный канал ПДП.
[7:6]	-	-	Зарезервировано
[5]	ЧТ/ЗП	dmaX_spr	Бит dmaX_spr устанавливает режим, в котором канал ПДП завершает свою работу при приёме короткого или пустого пакета (условие окончания передачи блока данных по USB).
[4]	ЧТ/ЗП	dmaX_int_empty	Бит dmaX_int_empty устанавливает, в какой момент устройство должно устанавливать статус завершения работы канала: 0 – в момент завершения операции с контроллером ПДП, 1 – в момент завершения операции с буфером конечной точки.
[3]	ЧТ/ЗП	dmaX_sendnull	Бит dmaX_sendnull устанавливает режим отправки пустого пакета в завершающей транзакции USB (работает, когда при передаче блока данных последний пакет получается максимального размера): 0 – пустой пакет не отправляется, 1 – отправляется. Обычно для завершения передачи блока данных по USB в таком случае необходимо отправлять пустой пакет.
[2]	ЧТ/ЗП	dmaX_mode	Бит dmaX_mode устанавливает режим работы с контроллером ПДП: значение 0 – контроллер USB устанавливает сигнал запроса не снимает его до получения сигнала подтверждения, значение 1 – контроллер USB устанавливает сигнал запроса на 1 такт.
[1]	ЧТ	dmaX_busy	Бит dmaX_busy показывает, что данный канал работает.
[0]	ЧТ/ЗП	dmaX_st	Запись значения 1 в бит dmaX_st производит запуск канала, запись значения 0 – остановку. Бит dmaX_st может сбрасывается автоматически, когда передача завершена или когда передача не может быть совершена (выключена соответствующая конечная точка или весь контроллер).

8.5.2.24 Регистры состояния ПДП DMAS1, DMAS2

Для каждого канала ПДП имеются регистры одинакового формата, представленного ниже (см. Таблица 8.31). Под символом X понимается 1 или 2 для соответствующего канала.

Таблица 8.31 - Формат регистров DMASX

Биты	Тип	Название поля	Описание
[31:2]	-	-	Зарезервировано
[1]	ЧТ	dmaX_sp	Поле dmaX_sp показывает размер последнего принятого или переданного пакета: 0 – последний пакет был максимального размера, 1 – последний пакет был коротким или пустым.
[0]	ЧТ	dmaX_np	Поле dmaX_np показывает размер последнего принятого или переданного пакета: 0 – последний пакет не был пустым, 1 – последний пакет был пустым.

8.5.2.25 Регистры счётчиков данных ПДП DMATCI1, DMATCI2

Для каждого канала ПДП имеются регистры одинакового формата, представленного ниже (см. Таблица 8.32). Под символом X понимается 1 или 2 для соответствующего канал.

Таблица 8.32 - Формат регистров DMATCIX

Биты	Тип	Название поля	Описание
[31:0]	ЧТ/ЗП	dmaX_tci	Поле dmaX_tci задаёт общий размер данных в байтах, которые следует передать за один запуск канала ПДП.

8.5.2.26 Регистры счётчиков данных ПДП DMATC1, DMATC2

Для каждого канала ПДП имеются регистры одинакового формата, представленного ниже (см. Таблица 8.33). Под символом X понимается 1 или 2 для соответствующего канала.

Таблица 8.33 - Формат регистров DMATCX

Биты	Тип	Название поля	Описание
[31:0]	ЧТ/ЗП	dmaX_tc	Поле dmaX_tci задаёт общий размер переданных за запуск канала ПДП данных в байтах.

8.5.2.27 Регистр управления конечной точкой 0 EPCTRL0

Формат регистра EPCTRL0 приведён ниже (см. Таблица 8.34).

Таблица 8.34 - Формат регистра EPCTRL0

Биты	Тип	Название поля	Описание
[31]	-	-	Зарезервировано
[30]	ЧТ	nack_int	Бит nack_int устанавливается при отправке пакета NAK. Для того чтобы сбросить данный бит, необходимо произвести запись в регистр EPCMD с установленным битом nack_int_clr.
[29]	ЧТ	stalled_int	Бит stalled_int устанавливается при отправке пакета STALL. Для того чтобы сбросить данный бит, необходимо произвести запись в регистр EPCMD с установленным битом stalled_int_clr.
[28]	ЧТ	ping_int	Бит ping_int устанавливается при получении пакета PING. Для того чтобы сбросить данный бит, необходимо произвести запись в регистр EPCMD с установленным битом ping_int_clr.
[27]	ЧТ	readyo_int	Бит readyo_int устанавливается в следующих случаях: выдача подтверждения ACK в ответ на транзакцию SETUP, бит fullo изменяет своё значение с 0 на 1. Для того чтобы сбросить данный бит, необходимо произвести запись в регистр EPCMD с установленным битом readyo_int_clr.
[26]	ЧТ	readyi_int	Бит readyi_int устанавливается в следующих случаях: выдача подтверждения ACK в ответ на транзакцию SETUP, бит emptyi изменяет своё значение с 0 на 1. Для того чтобы сбросить данный бит, необходимо произвести запись в регистр EPCMD с установленным битом readyi_int_clr.
[25:23]	-	-	Зарезервировано
[22]	ЧТ	nack_inten	Биты nack_inten, stalled_inten, ping_inten, readyo_inten, readyi_inten разрешают запрос на прерывание по установке соответствующих битов.
[21]	ЧТ	stalled_inten	
[20]	ЧТ	ping_inten	
[19]	ЧТ	readyo_int_en	
[18]	ЧТ	readyi_int_en	
[17:14]	-	-	Зарезервировано
[13]	ЧТ	toggle	Бит toggle покаывает текущее значение секвенсора.
[12]	ЧТ	stall	Бит stall показывает, находится ли конечная точка в остановленном состоянии (Stalled State).
[11]	ЧТ	fullo	Бит fullo показывает наличие данных в буфере конечной точки типа OUT.
[10]	ЧТ	emptyo	Бит emptyo показывает отсутствие данных в буфере конечной точки типа OUT.
[9]	ЧТ	fulli	Бит fulli показывает наличие данных в буфере конечной точки типа IN.
[8]	ЧТ	emptyi	Бит emptyi показывает отсутствие данных в буфере конечной точки типа IN.
[7:1]	-	-	Зарезервировано
[0]	ЧТ	ep_en	Бит ep_en показывает, включена ли данная конечная точка.

8.5.2.28 Регистры управления конечными точками 1-15 EPCTRL1-EPCTRL15

Формат регистра EPCTRL приведён ниже (см. Таблица 8.35).

Таблица 8.35 - Формат регистра EPCTRL

Биты	Тип	Название поля	Описание
[31]	-	-	Зарезервировано
[30]	ЧТ	nack_int	Бит nack_int устанавливается при отправке пакета NAK. Для того чтобы сбросить данный бит, необходимо произвести запись в регистр EPCMD с установленным битом nack_int_clr.
[29]	ЧТ	stalled_int	Бит stalled_int устанавливается при отправке пакета STALL. Для того чтобы сбросить данный бит, необходимо произвести запись в регистр EPCMD с установленным битом stalled_int_clr.

Продолжение таблицы 8.35

Биты	Тип	Название поля	Описание
[28]	ЧТ	ping_int	Бит ping_int устанавливается при получении пакета PING. Для того чтобы сбросить данный бит, необходимо произвести запись в регистр EPCMD с установленным битом ping_int_clr.
[27]	ЧТ	empty_int	Бит empty_int устанавливается, когда буфер становится пустым. Для того чтобы сбросить данный бит, необходимо произвести запись в регистр EPCMD с установленным битом empty_int_clr.
[26]	ЧТ	ready_int	Бит ready_int устанавливается в следующих случаях: для конечных точек типа IN – бит full изменяет своё значение с 0 на 1, для конечных точек тип OUT – бит empty изменяет своё значение с 0 на 1. Для того чтобы сбросить данный бит, необходимо произвести запись в регистр EPCMD с установленным битом ready_int_clr.
[25:23]	-	-	Зарезервировано
[22]	ЧТ	nack_inten	Биты nack_inten, stalled_inten, ping_inten, empty_inten, ready_inten разрешают запрос на прерывание по установке соответствующих битов.
[21]	ЧТ	stalled_inten	
[20]	ЧТ	ping_inten	
[19]	ЧТ	empty_inten	
[18]	ЧТ	ready_inten	
[17]	ЧТ	nackresp	Бит nackresp показывает, отвечает ли данная конечная точка пакетом NAK на пакеты IN, OUT, PING.
[16]	ЧТ	nullresp	Бит nullresp показывает, отвечает ли данная конечная точка пустым пакетом на транзакции IN типа Bulk или Interrupt.
[15:14]	ЧТ	hiband	Поле hiband показывает количество пакетов, отправляемое за 1 (микро)фрейм.
[13]	ЧТ	toggle	Бит toggle покаывает текущее значение секвенсора.
[12]	ЧТ	stall	Бит stall показывает, находится ли конечная точка в остановленном состоянии (Stalled State).
[11]	ЧТ	full	Бит full показывает наличие данных в буфере конечной точки.
[10]	ЧТ	empty	Бит empty показывает отсутствие данных в буфере конечной точки.
[9:8]	ЧТ	phyptr	Поле phyptr показывает номер буфера (00 или 01), к которому в данный момент производится доступ со стороны USB.
[7:6]	ЧТ	appptr	Поле appptr показывает номер буфера (00 или 01), к которому в данный момент производится доступ со стороны процессорной системы.
[5:4]	ЧТ	bnum	Поле bnum показывает тип буферизации: 00 – одинарная, 01 – двойная, 10, 11 – зарезервировано (не допускается устанавливать это значение).
[3]	ЧТ	dir	Бит dir показывает направление передачи данных для данной конечной точки: 0 – OUT, 1 – IN.
[2:1]	ЧТ	et	Поле et показывает тип транзакции: 00 – запрещенный код, 01 – Isochronous, 10 – Bulk, 11 – Interrupt.
[0]	ЧТ	ep_en	Бит ep_en показывает, включена ли данная конечная точка.

8.5.2.29 Регистр конфигурации конечных точек EPCONF0-EPCONF15

Формат регистра EPCONF приведён ниже (см. Таблица 8.36).

Таблица 8.36 - Формат регистра EPCONF

Биты	Тип	Название поля	Описание
[31:29]	-	-	Зарезервировано
[28:24]	ЧТ/ЗП	countidx	Поле countidx устанавливает номер (от 0 до 31) счётчика данных (регистр EPCOUNT) для данной конечной точки.
[23:13]	ЧТ/ЗП	size	Поле size устанавливает размер буфера в байтах для данной конечной точки. Максимальный размер пакета, передаваемого данной конечной точкой, равен размеру этого буфера. Допустимые значения: в режиме full-speed для конечных точек типа Control – 8, 16, 32 или 64 байта, в режиме full-speed для конечных точек типа Bulk – 8, 16, 32 или 64 байта, в режиме full-speed для конечных точек типа Interrupt – от 1 до 64 байт, в режиме full-speed для конечных точек типа Isochronous – от 1 до 1024 байт, в режиме high-speed для конечных точек типа Control – 64 байта, в режиме high-speed для конечных точек типа Bulk – 512 байт, в режиме high-speed для конечных точек типа Interrupt – от 1 до 1024 байт, в режиме high-speed для конечных точек типа Isochronous – от 1 до 1024 байт.
[12:0]	ЧТ/ЗП	base	Поле base задаёт начальный адрес буфера конечной точки во внутренней памяти контроллера. От физического адреса в байтах нужно отбросить 2 младших бита и 1 старший. Например, адрес – 8180h, поле base – 0060h; адрес – FF00h, поле base – 1FC0h.

8.5.2.30 Регистр счётчиков конечных точек EPCOUNT0-EPCOUNT31

Формат регистра EPCOUNT приведён ниже (см. Таблица 8.37).

Таблица 8.37 - Формат регистра EPCOUNT

Биты	Тип	Название поля	Описание
[31:27]	-	-	Зарезервировано
[26:16]	ЧТ	phycnt	Поле phycnt показывает текущее состояние счётчика данных, переданных со стороны физического интерфейса. Поле обновляется вместе с отправкой подтверждения, сбрасывается в начале приёма пакета.
[15:11]	-	-	Зарезервировано
[10:0]	ЧТ	appcnt	Поле appcnt показывает текущее состояние счётчика данных, записанных или прочитанных в буфер конечной точки.

8.5.2.31 Регистры управления каналами хост-контроллера HCEPCTRL1, HCEPCTRL2

Формат регистра HCEPCTRL1 приведён ниже (см. Таблица 8.38).

Таблица 8.38 - Формат регистра HCEPCTRL1

Биты	Тип	Название поля	Описание
[31]	ЧТ	nlinkinvalid	Поле nlinkinvalid показывает, следует ли контроллеру перейти к выполнению передачи в следующем канале. 0 – запланированных передач больше нет, 1 – перейти к выполнению передачи канала с номером nextlink.
[30:28]	ЧТ	nextlink	Поле nextlink показывает номер канала, к выполнению передачи в котором следует приступить после выполнения передачи в данном канале.
[27:26]	ЧТ	speed	Поле speed показывает скорость канала: 00 – full speed, 01 – low speed, 10 – high speed, 11 – запрещенный код.
[25:24]	ЧТ	sendpid	Поле sendpid показывает тип транзакции: 00 – OUT, 01 – IN, 10 – SETUP, 11 – PING. При использовании механизма PING поле автоматически меняет тип с PING на OUT и обратно в соответствии с ответом конечного устройства.
[23:22]	-	-	Зарезервировано
[21:20]	ЧТ	nyetcnt	Поле nyetcnt считает количество принятых подряд пакетов NYET или MDATA в транзакциях Complete-Split. Условия инкремента: при транзакции Complete-Split Interrupt IN принят пакет NYET или MDATA, при транзакции Complete-Split Interrupt OUT принят пакет NYET. Условия сброса: при транзакции Start-Split, при приёме подтверждения ACK на транзакцию Complete-Split Interrupt OUT.

Продолжение таблицы 8.38

Биты	Тип	Название поля	Описание
[19:16]	ЧТ	status	Поле status показывает результат транзакции. Бит 16 – Halt (ошибка более трёх раз). Бит 17 – Stall (конечное устройство ответило пакетом STALL). Бит 18 – пропущен микрофрейм (для транзакции был установлен бит fmtsel, но она попала в другой микрофрейм). Бит 19 зарезервирован.
[15:14]	ЧТ	errcnt	Поле errcnt показывает текущее состояние счётчика ошибок. При запуске передачи следует устанавливать значение 0, а когда количество ошибок превышает 3, то аппаратно снимается бит trans_en. Поле инкрементируется всегда, когда возникает необходимость повторной пересылки данных, кроме случаев, когда работает счётчик nyetcnt.
[13]	ЧТ	toggle	Бит toggle показывает текущее значение секвенсора данных.
[12]	ЧТ	sc	Поле sc задаёт тип пакета SPLIT (если используется): значение 0 – Start-Split, значение 1 – Complete-Split.
[11]	ЧТ	full	Бит full показывает, что буфер канала содержит данные.
[10]	ЧТ	empty	Бит empty показывает, что буфер канала не содержит данных.
[9:8]	ЧТ	hcptr	Поле hcptr показывает номер буфера (00 или 01), к которому в данный момент производится доступ со стороны USB.
[7:6]	ЧТ	appptr	Поле appptr показывает номер буфера (00 или 01), к которому в данный момент производится доступ со стороны процессорной системы
[5:4]	ЧТ	bnum	Поле bnum показывает тип буферизации: 00 – одинарная, 01 – двойная, 10, 11 – запрещенный код.
[3]	-	-	Зарезервировано
[2:1]	ЧТ	et	Поле et задаёт тип транзакции USB: 00 – Control, 01 – Isochronous, 10 – Bulk, 11 – Interrupt.
[0]	ЧТ	trans_en	Поле trans_en, работает ли данный канал. Запуск и остановка осуществляется с помощью регистра EPCMD.

Формат регистра HCEPCTRL2 приведён ниже (см. Таблица 8.39)

Таблица 8.39 - Формат регистра HCEPCTRL2

Биты	Тип	Название поля	Описание
[31]	ЧТ/ЗП	intervalsel	Бит intervalsel задаёт единицы, в которых задан период передач типа Isochronous и Interrupt, если выбран циклический режим (fmtsel=0). Сам период задаётся в поле interval. Значение 0 бита intervalsel выбирает в качестве единицы периода один микрофрейм, значение 1 – один фрейм.
[30]	ЧТ/ЗП	fmtsel	Бит fmtsel задаёт режим запуска передач типа Isochronous и Interrupt. Значение 0 – циклический, значение 1 – по совпадению номера фрейма.
[29:20]	ЧТ/ЗП	interval	Поле interval задаёт период запуска передач типа Isochronous и Interrupt. В циклическом режиме (fmtsel=0) и при выборе единицы в один микрофрейм (intervalsel=0): 000 – передача каждый 1 микрофрейм, 001 – передача каждые 2 микрофрейма, 010 – передача каждые 4 микрофрейма, 011 – передача каждые 8 микрофреймов, В циклическом режиме (fmtsel=0) и при выборе единицы в один фрейм (intervalsel=1): 000 – передача каждый 1 фрейм, 001 – передача каждые 2 фрейма, 010 – передача каждые 4 фрейма, 011 – передача каждые 8 фреймов, 100 – передача каждые 16 фреймов, 101 – передача каждые 32 фрейма, 110 – передача каждые 64 фрейма, 111 – передача каждые 128 фреймов, В режиме по совпадению номера фрейма (fmtsel=1) и в режиме high-speed: каждый микрофрейм, в котором поле HCFRMIDX[9:0] совпадает с полем interval. В режиме по совпадению номера фрейма (fmtsel=1) и в режиме full-speed или low-speed: каждый микрофрейм, в котором поле HCFRMIDX[9:3] совпадает с полем interval[9:3].
[19:16]	ЧТ/ЗП	funcaddr	Поле funcaddr задаёт адрес конечного устройства, с которым производится транзакция. Можно задать номер от 0 до 7, так как данный хост-контроллер поддерживает только 8 устройств.
[15:12]	ЧТ/ЗП	endpnumber	Поле endpnumber задаёт адрес конечной точки, с которой производится транзакция. Значение от 0 до 15.
[11:8]	ЧТ/ЗП	hubaddr	Поле hubaddr задаёт адрес концентратора, к которому подключено устройство, с которым производится транзакция. Допустимые значения – от 0 до 15.
[7:4]	ЧТ/ЗП	hubportnumber	Поле hubportnumber задаёт номер порта концентратора, к которому подключено устройство, с которым производится транзакция. Допустимые значения – от 0 до 15.
[3:2]	-	-	Зарезервировано
[1]	ЧТ/ЗП	startbit	Бит startbit устанавливет бит S в пакетах SPLIT.
[0]	ЧТ/ЗП	endbit	Бит endbit устанавливает бит E (для Start-Split) и U (для Complete-Split) в пакетах типа SPLIT.

8.5.2.32 Регистр конфигурации каналов хост-контроллера HCEPCONF0-HCEPCONF7

Формат регистра HCEPCONF приведён ниже (см. Таблица 8.40).

Таблица 8.40 - Формат регистра HCEPCONF

Биты	Тип	Название поля	Описание
[31:29]	-	-	Зарезервировано
[28:24]	ЧТ/ЗП	countidx	Поле countidx устанавливает номер (от 0 до 31) счётчика данных (регистр HCEPCOUNT) для данного канала.
[23:13]	ЧТ/ЗП	size	Поле size устанавливает размер буфера в байтах для данного канала. Максимальный размер пакета, передаваемого данным каналом, равен размеру этого буфера. Допустимые значения: в режиме low-speed для каналов Control – 8 байт, в режиме low-speed для каналов Interrupt – от 1 до 8 байт, в режиме full-speed для каналов Control – 8, 16, 32 или 64 байта, в режиме full-speed для каналов Bulk – 8, 16, 32 или 64 байта, в режиме full-speed для каналов Interrupt – от 1 до 64 байт, в режиме full-speed для каналов Isochronous – от 1 до 1024 байт, в режиме high-speed для каналов Control – 64 байта, в режиме high-speed для каналов Bulk – 512 байт, в режиме high-speed для каналов Interrupt – от 1 до 1024 байт, в режиме high-speed для каналов Isochronous – от 1 до 1024 байт.
[12:0]	ЧТ/ЗП	base	Поле base задаёт начальный адрес буфера канала во внутренней памяти контроллера. От физического адреса в байтах нужно отбросить 2 младших бита и 1 старший. Например, адрес – 8180h, поле base – 0060h; адрес – FF00h, поле base – 1FC0h.

8.5.2.33 Регистр счётчиков каналов хост-контроллера HCEPCOUNT0-HCEPCOUNT31

Формат регистра HCEPCOUNT приведён ниже (см. Таблица 8.41).

Таблица 8.41 - Формат регистра HCEPCOUNT

Биты	Тип	Название поля	Описание
[31:30]	ЧТ	rdatapid	Поле rdatapid показывает идентификатор пакета полученных данных (последней успешной транзакции): 00 – DATA0, 01 – DATA1, 10 – DATA2, 11 – MDATA.
[29:27]	-	-	Зарезервировано
[26:16]	ЧТ	hccnt	Поле hccnt показывает текущее состояние счётчика данных, переданных со стороны физического интерфейса. Поле обновляется вместе с отправкой подтверждения, сбрасывается в начале приёма пакета.
[15:11]	-	-	Зарезервировано
[10:0]	ЧТ/ЗП	appcnt	Поле appcnt показывает текущее состояние счётчика данных, записанных или прочитанных в буфер канала.

8.6 Контроллер начальной загрузки BROMC

Контроллер начальной загрузки обеспечивает выполнение процедуры начальной инициализации СБИС после системного сброса.

Контроллер содержит в своем составе ПЗУ начальной загрузки объемом 1024*32 разрядных слова, в котором хранится программа начальной инициализации микросхемы и конечный автомат, обеспечивающий запись содержимого данного ПЗУ во внутреннюю память одной из процессорных систем по нулевым адресам.

Выбор процессорной системы, которая выполняет процедуру начальной инициализации СБИС, определяется состоянием конфигурационного вывода BOOTM2:

0 – NMPU0;
1 – NMPU1.

По сигналу системного сброса адресные регистры генератора выборки команд обоих процессорных систем устанавливаются в нулевые значения (указывают на нулевой адрес внутренней памяти). Выборка команд при этом заблокирована.

После снятия сигнала системного сброса, в зависимости от состояния вывода BOOTM2, контроллер запускает процедуру ПДП, которая помещает содержимое ПЗУ начальной загрузки по нулевым адресам внутренней памяти выбранной процессорной системы. После окончания процедуры ПДП контроллер разрешает выборку команд для данной процессорной системы.

После этого процессорное ядро начинает выполнение программы инициализации.

8.7 Контроллер ПДП память-периферия KDMAC

Контроллер KDMAC служит для пересылки данных из периферийных устройств в память и обратно. Доступ к памяти осуществляется с использованием 32-рязрядной шины AXI. Доступ к периферии осуществляется с использованием другой 32-рязрядной шины AXI. Программирование KDMAC осуществляется по шине APB. В данном случае KDMAC поддерживает работу только с контроллером SPI, поскольку контроллер USB имеет свой встроенный контроллер ПДП.

8.7.1 Программно доступные регистры контроллера KDMAC

Контроллер KDMAC имеет 2 канала. Один из них предназначен для пересылок данных типа память-периферия, другой – для пересылок типа периферия – память. К каждому каналу относятся 8 программно доступных регистров. Для доступа к этим регистрам у контроллера имеется шина APB. Программно доступные регистры KDMAC расположены в области памяти периферийных устройств процессора, и имеют начальный адрес 0x1003_1800h и общий размер 4 Кбайт. В Таблица 8.42 дан список всех программно доступных регистров KDMAC, а в Таблица 8.43 представлено их функциональное назначение.

Для того, чтобы контроллер KDMAC начал пересылку данных, необходимо произвести инициализацию канала и записать единичку в младший бит регистра Enable. Инициализация канала состоит в задании адреса источника, адреса приемника, числа байтов, которые необходимо передать, типа упаковки/распаковки данных и типа сигнала peripheral_request.

Фактически тип упаковки/распаковки данных обозначает ту разрядность данных, с которой может работать соответствующее периферийное устройство. Работа с памятью всегда происходит 32-разрядными словами. Поддерживаются два типа упаковки/распаковки данных: по 8 разрядов и по 16 разрядов. Для правильной работы KDMAC необходима также соответствующая настройка контроллера SPI (в нём также есть два режима упаковки/распаковки данных).

Преимуществом 8-разрядного режима упаковки/распаковки является то, что можно передавать нечётное количество байтов, даже один байт.

Преимущество 16-разрядного режима и том, что этот режим в 2 раза меньше нагружает периферийную шину системы (это может быть важно на максимальной скорости SPI, если необходимо обеспечить непрерывную команду чтения/записи по SPI).

После пересылки каналом заданного числа байтов автоматически устанавливается 1 в регистре прерывания и на контроллер прерываний обеих процессорных систем выдается прерывание. Для того, чтобы снова запустить канал, необходимо сбросить в 0 регистр прерывания. Прерывание тут же автоматически сбросится.

Для программного останова канала необходимо сбросить регистр запуска/останова канала в 0. При программном останове канала регистр прерывания в единицу не устанавливается. Чтобы начать новую передачу после программного останова, необходимо подождать, пока регистр занятости канала сбросится в 0. Это произойдет сразу же, как только закончится передача текущего 32-битного слова.

Во время работы канала (Busy = 1) запись в регистры не будет иметь результата, за исключением записи 0 в регистр запуска/останова канала.

Таблица 8.42 - Программно доступные регистры контроллера KDMAC

Номер канала	Имя регистра	Адрес	Тип доступа	Описание
1 FtS	SrcAdr_FtS1	0x1003_1800h	R/W	Адрес источника
	DstAdr_FtS1	0x1003_1801h	R/W	Адрес приемника
	TrSize_FtS1	0x1003_1802h	R/W	Число передаваемых байт данных
	Enable_FtS1	0x1003_1803h	R/W	Запуск/останов канала
	Interrupt_FtS1	0x1003_1804h	R/W	Отображает состояние прерывания. Записать можно только 0.
	Busy_FtS1	0x1003_1805h	RO	Занятость канала
	PackType_FtS1	0x1003_1806h	R/W	Тип упаковки данных
	PerHandsh_FtS_1	0x1003_1807h	R/W	Тип сигнала peripheral_request
1 StF	SrcAdr_StF1	0x1003_1880h	R/W	Адрес источника
	DstAdr_StF1	0x1003_1881h	R/W	Адрес приемника
	TrSize_StF1	0x1003_1882h	R/W	Число передаваемых байт данных
	Enable_StF1	0x1003_1883h	R/W	Запуск/останов канала
	Interrupt_StF1	0x1003_1884h	R/W	Отображает состояние прерывания. Записать можно только 0.
	Busy_StF1	0x1003_1885h	RO	Занятость канала
	PackType_StF1	0x1003_1886h	R/W	Тип распаковки данных
	PerHandsh_StF_1	0x1003_1887h	R/W	Тип сигнала peripheral_request

Таблица 8.43 - Функциональное назначение регистров контроллера KDMAC

Название регистра	Значимая часть	Начальное значение	Функциональное назначение
SrcAdr_FtS1	Все 32 бита	00000000h	Адрес источника. Записываемое значение адреса должно быть выровнено до 32-разрядного слова, т. е. два младших бита должны быть нулевыми. Запись возможна только когда канал не занят (младший бит регистра Busy_FtS1 равен нулю). Во время работы канала происходит инкрементирование адреса.
DstAdr_FtS1	Все 32 бита	00000000h	Адрес приемника. Записываемое значение адреса должно быть выровнено до 32-разрядного слова, т. е. два младших бита должны быть нулевыми. Запись возможна только когда канал не занят (младший бит регистра Busy_FtS1 равен нулю). Инкрементирование адреса во время работы канала не происходит.
TrSize_FtS_1	Младшие 24 бита	00001h	Число передаваемых байт данных. Запись возможна только когда канал не занят (младший бит регистра Busy_FtS1 равен нулю). Во время чтения выдает оценочное значение, сколько байтов осталось еще передать.
Enable_FtS_1	Младший бит	0	Регистр запуска/останова канала. Для запуска канала необходимо записать значение 1. Для программного останова канала необходимо записать значение 0. Повторный запуск канала возможен только после завершения каналом текущей передачи 32-разрядного слова. Потери данных во внутреннем FIFO не происходит, т. к. перед остановом канала все уже считанные данные пересылаются.
Interrupt_FtS_1	Младший бит	0	Регистр прерывания. После завершения работы канала по причине пересылки каналом всех необходимых данных устанавливается в 1. Если причиной завершения работы канала был программный останов, то значение регистра остается 0. Чтобы снова запустить канал, необходимо записать значение 0 в младший бит регистра прерывания. Записать в данный регистр возможно только 0.
Busy_FtS_1	Младший бит	0	Регистр, показывающий занятость канала: 0 – канал свободен; 1 – канал занят; Основное отличие этого регистра от регистра Enable_FtS_1 состоит в том, что при программном останове, когда в регистр Enable_FtS_1 записывается значение 0, регистр Busy_FtS сбрасывается в 0 только после завершения передачи текущей 32-разрядной порции данных.

Продолжение таблицы 8.43

Название регистра	Значимая часть	Начальное значение	Функциональное назначение
PackType_FtS_1	Младшие 2 бита	10b	Задаёт тип упакованных данных. Следует задать одно из двух значений: 01b – 8-разрядные слова; 10b – 16-разрядные слова.
PerHandsh_FtS_1	Младший бит	0	0 – сигнал Peripheral_request соответствует стандартной спецификации; 1 – сигнал Peripheral_request соответствует устройству UART. Запись возможна только когда канал не занят (младший бит регистра Busy_FtS1 равен нулю).
SrcAdr_StF1	Все 32 бита	00000000h	Адрес источника. Записываемое значение адреса должно быть выровнено до 32-разрядного слова, т.е. два младших бита должны быть нулевыми. Запись возможна только когда канал не занят (младший бит регистра Busy_FtS1 равен нулю Инкрементирование адреса во время работы канала не происходит.
DstAdr_StF1	Все 32 бита	00000000h	Адрес приемника. Записываемое значение адреса должно быть выровнено до 32-разрядного слова, т.е. три младших бита должны быть нулевыми. Запись возможна только когда канал не занят (младший бит регистра Busy_FtS1 равен нулю). Во время работы канала происходит инкрементирование адреса.
TrSize_StF_1	Младшие 24 бита	00001h	Число передаваемых байт данных. Запись возможна только когда канал не занят (младший бит регистра Busy_FtS1 равен нулю). Во время чтения выдает оценочное значение, сколько байтов осталось еще передать.
Enable_StF_1	Младший бит	0	Регистр запуска/останова канала. Для запуска канала необходимо записать значение 1. Для программного останова канала необходимо записать значение 0. Повторный запуск канала возможен только после завершения каналом текущей передачи 32-разрядного слова. Потери данных во внутреннем FIFO не происходит, т.к. перед остановом канал завершает передачу текущей 32-разрядной порции данных.
Interrupt_StF_1	Младший бит	0	Регистр прерывания. После завершения работы канала по причине пересылки каналом всех необходимых данных устанавливается в 1. Если причиной завершения работы канала был программный останов, то значение регистра остается 0. Чтобы снова запустить канал необходимо записать значение 0 в младший бит регистра прерывания. Записать в данный регистр возможно только 0.

Продолжение таблицы 8.43

Название регистра	Значимая часть	Начальное значение	Функциональное назначение
Busy_StF_1	Младший бит	0	Регистр, показывающий занятость канала. 0 – канал свободен; 1 – канал занят. Основное отличие этого регистра от регистра Enable_FtS_1 состоит в том, что при программном останове, когда в регистр Enable_FtS_1 записывается значение 0 с шины APB, регистр Busy_FtS сбрасывается в 0 только после завершения передачи текущей 32-разрядной порции данных.
PackType_StF_1	Младшие 2 бита	10b	Задаёт тип упакованных данных. Следует задать одно из двух значений: 01b – 8-разрядные слова; 10b – 16-разрядные слова.
PerHandsh_StF_1	Младший бит	0	0 – сигнал Peripheral_request соответствует стандартной спецификации; 1 – сигнал Peripheral_request соответствует устройству UART. Запись возможна только когда канал не занят (младший бит регистра Busy_FtS1 равен нулю).

8.7.2 Описание работы контроллера KDMAC

8.7.2.1 Канал FtS

После инициализации и запуска канала контроллер KDMAC ждет прихода от периферийного устройства запроса на передачу данных (periph_request = 1). Сразу же после получения сигнала начинается чтение из памяти 32-разрядных данных по шине AXI. Затем происходит запись данных требуемой разрядности (разрядность задается в регистре PackType_FtS_1) в соответствующий регистр периферийного устройства. После этого к периферийному устройству выдается подтверждение, что данные были переданы (periph_clr = 1).

Далее после каждого прихода запроса контроллер KDMAC записывает в соответствующий регистр периферийного устройства данные и выдает подтверждение передачи. Так происходит до тех пор, пока не будут переданы все байты 32-разрядного регистра, в котором находятся данные, считанные из памяти. Затем при приходе следующего запроса сначала будет считано 32-разрядное слово данных из памяти и потом записаны данные требуемой разрядности в периферийное устройство.

Как только будет передано заданное количество байт, канал прекращает свою работу и выдает прерывание.

Если во время работы канала произошел программный останов, то полностью передается уже считанное из памяти 32-разрядное слово данных, и только потом канал останавливает свою работу. Прерывание при этом не выдается.

8.7.2.2 Канал StF

После инициализации и запуска канала контроллер KDMAC ждет прихода от периферийного устройства запроса на передачу данных (periph_request = 1). Сразу же после получения запроса начинается чтение данных требуемой разрядности (разрядность задается в регистре PackType_StF_1) из соответствующего регистра периферийного устройства. После этого периферийному устройству выдается подтверждение, что данные были считаны (periph_clr = 1).

Далее после каждого прихода запроса контроллер ПДП читает из соответствующего регистра периферийного устройства данные и выдает подтверждение чтения. Так происходит до тех пор, пока не заполнится полностью 32-разрядный регистр данных контроллера ПДП. Как только это происходит, сразу же начинается запись этого слова данных в память по шине.

Затем опять происходит заполнение 32-разрядного регистра.

Как только будет передано заданное количество байт, канал прекращает свою работу и выдает прерывание.

Если во время работы канала произошел программный останов, то происходит заполнение 32-разрядного регистра, он записывается в память, и только потом канал останавливает свою работу. Прерывание при этом не выдается.

8.7.3 Особенности работы контроллера KDMAC

Контроллер ПДП всегда работает с памятью 32-разрядными словами.

Если происходит передача данных из памяти в периферию и количество передаваемых байт не кратно 32-разрядному слову, то происходит передача только необходимого числа 16- или 8-разрядных данных.

В режиме упаковки 16-разрядных данных не следует задавать нечётное значение TrSize.

Если происходит передача данных из периферии в память и количество передаваемых байтов не составляет полного 32-разрядного слова, то неактуальные старшие разряды заполняются нулями.

Если младший бит регистра Interrupt равен 1, то блокируется запись лишь регистра Enable. Во все остальные регистры соответствующего канала возможно записать необходимые значения. Таким образом, происходит защита от повторного запуска канала до снятия бита прерывания, но при этом возможно делать предварительные настройки канала.

После завершения работы канала регистры не возвращаются в состояния до запуска канала. Это относится к регистрам SrcAdr, DstAdr, TrSize. Они остаются в том состоянии, до которого дошли к концу работы канала. Таким образом, нужно задать новые значения для этих регистров при запуске новой передачи.

При программном останове канал завершает свою работу только после сброса в 0 регистра Busy. Канал не может остановиться сразу, т. к. происходит завершение пересылки того 32-разрядного слова, которое уже начало передаваться. Это делается во избежание потери данных во внутренних регистрах контроллера и для того, чтобы пользователь имел возможность продолжить пересылку данных, которую по некоторой причине пришлось приостановить. Поэтому перед тем, как дальше работать с каналом, необходимо сначала проверить, равен ли нулю бит Busy или нет. И только когда этот бит сбросится в ноль, можно начать использовать данный канал.

8.8 Сторожевой таймер WDT

Сторожевой таймер служит индикатором попадания процессора в состояние “зависания” и сброса процессора в случае обнаружения данной ситуации. Управление блоком сторожевого таймера осуществляется по шине AMBA APB версии 2.0. Блок состоит из программируемого 32-разрядного таймера, работающего на тактовой частоте шины APB, схемы управления режимами работы таймера и схемы формирования контрольных и управляющих сигналов.

8.8.1 Организация работы сторожевого таймера

Временная диаграмма работы блока сторожевого таймера представлена на Рисунок 8.1.

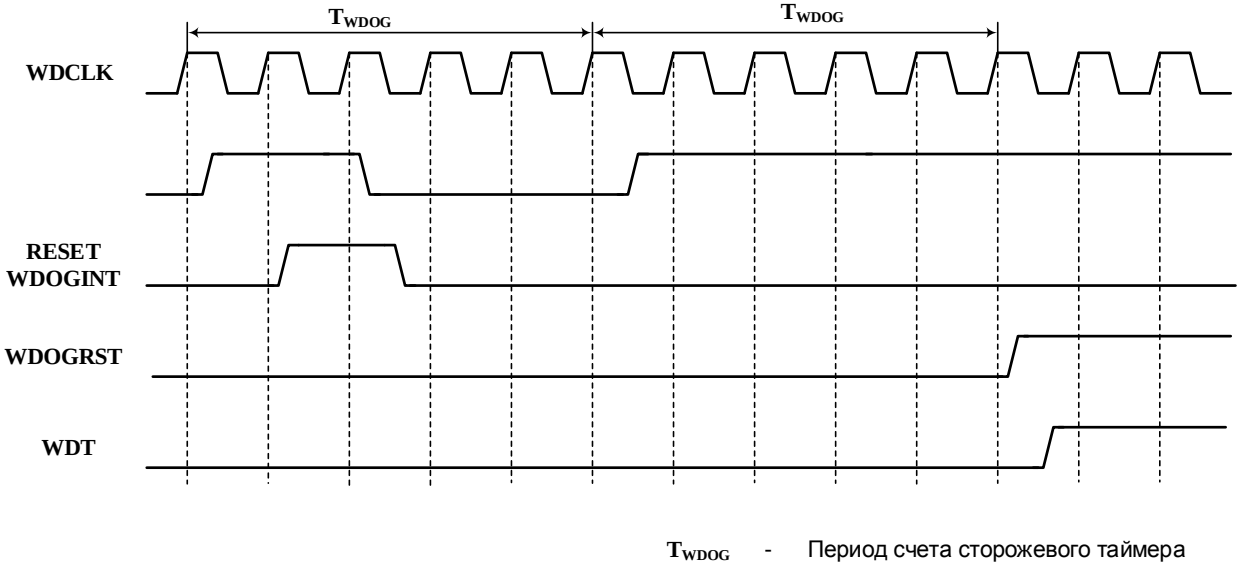


Рисунок 8.1 – Временные диаграммы работы сторожевого таймера

После снятия сигнала системного сброса, блок находится в выключенном состоянии. Если возникла необходимость воспользоваться сторожевым таймером, то программным образом надо задать интервал счета таймера, формируя тем самым квант времени работы сторожевого таймера. После записи интервала счета счетчик таймера начинает декрементироваться.

Когда счетчик достигает нулевого значения, генерируется сигнал прерывания WDOGINT. Данный сигнал прерывания заводится на контроллеры прерываний обоих процессорных систем. Одновременно в счетчик заносится начальное значение интервала счета и декремент счетчика возобновляется. Если до достижения счетчиком нулевого значения прерывание WDOGINT не будет обработано (сброшено), то будет сформирован сигнал WDOGRST. Данный сигнал выводится на внешние вывод WDT.

Для исключения случайного сброса процессора вследствие неправильной записи в регистры блока сторожевого таймера предусмотрена возможность программно заблокировать эти регистры для записи.

8.8.2 Программно доступные регистры блока сторожевого таймера

Программно доступные регистры блока сторожевого таймера WDT расположены в области памяти периферийных устройств процессора, и имеют начальный адрес WDT Base = 0x1003_0800h и общий размер 4 Кбайт. Спецификация регистров блока представлена в Таблица 8.44 .

Таблица 8.44 - Спецификация регистров блока WDT

Адрес	Тип	Разряд-ность	Начальное значение	Имя	Описание
WDT Base + 0x00	ЧТ/ЗП	32	0xFFFFFFFF	WdogLoad	Регистр загрузки. WdogLoad
WDT Base + 0x01	ЧТ	32	0xFFFFFFFF	WdogValue	Регистр значения. WdogValue
WDT Base + 0x02	ЧТ/ЗП	2	0x0	WdogControl	Регистр управления. WdogControl
WDOG Base + 0x03	ЗП	-	-	WdogIntClr	Регистр сброса прерываний. WdogIntClr
WDT Base + 0x04	ЧТ	1	0x0	WdogRIS	Регистр состояния прерывания до наложения маски. WdogRIS
WDT Base + 0x05	ЧТ	1	0x0	WdogMIS	Регистр состояния прерывания после наложения маски. WdogMIS
WDT Base + 0x06 to 0x2FF	-	-	-	-	Зарезервировано
WDT Base + 0x300	ЧТ/ЗП	32	0x0	WdogLock	Регистр блокировки. WdogLock
WDT Base + 0x301 to 0x3BF	-	-	-	-	Зарезервировано
WDT Base + 0x3C0	ЧТ/ЗП	1	0x0	WdogITCR	Регистр управления тестом интеграции. WdogITCR
WDT Base + 0x3C1	ЗП	2	0x0	WdogITOP	See Integration Test Output Set Register, WdogITOP
WDT Base + 0x3C2 to 0x3F7	-	-	-	-	Зарезервировано
WDT Base + 0x3F8	ЧТ	8	0x05	WdogPeriphID0	Идентификатор периферийного устройства. WdogPeriphID0
WDOG Base + 0x3F9	ЧТ	8	0x18	WdogPeriphID1	Идентификатор периферийного устройства. WdogPeriphID1
WDT Base + 0x3FA	ЧТ	8	0x14	WdogPeriphID2	Идентификатор периферийного устройства. WdogPeriphID2
WDT Base + 0x3FB	ЧТ	8	0x00	WdogPeriphID3	Идентификатор периферийного устройства. WdogPeriphID3
WDT Base + 0x3FC	ЧТ	8	0x0D	WdogPCellID0	Идентификатор PrimeCell. WdogPCellID0
WDT Base + 0x3FD	ЧТ	8	0xF0	WdogPCellID1	Идентификатор PrimeCell. WdogPCellID1
WDT Base + 0x3FE	ЧТ	8	0x05	WdogPCellID2	Идентификатор PrimeCell. WdogPCellID2
WDT Base + 0x3FF	ЧТ	8	0xB1	WdogPCellID3	Идентификатор PrimeCell. WdogPCellID3

WdogIntClr - регистр сброса прерываний. Любая запись в данный регистр снимает прерывание WDOG и перезагружает счетчик значением регистра WdogLoad.

WdogLoad - регистр загрузки - 32-битный регистр, доступный по записи и по чтению. Данный регистр хранит значение интервала счета, с которого счетчик начинает декрементироваться. Если значение регистра WdogLoad установлено в нуль, то прерывания сторожевым таймером вырабатываются незамедлительно. Если содержимое регистра отлично от нуля, то это значение загружается в счетчик незамедлительно после записи регистра.

WdogValue – регистр значения счетчика - 32-битный регистр, доступный только на чтение. При чтении этого регистра выдается текущее значение декрементирующего счетчика.

WdogControl - регистр управления - 2-битный регистр, доступный на запись и чтение, который разрешает программному обеспечению (ПО) управлять сторожевым таймером. В Таблица 8.45 приведен формат регистра WdogControl.

WdogIntClr - регистр сброса прерываний. Любая запись в данный регистр снимает прерывание WDOG и перезагружает счетчик значением регистра WdogLoad.

Таблица 8.45 - Формат регистра WdogControl

Биты	Название	Тип	Выполняемая функция
[31:2]	-	-	Зарезервировано
[1]	RESEN	ЧТ/ЗП	Разрешающий сигнал блока WDOG для сброса выхода, WDOGRST. Работает как маска для сброса выхода. Если установлен высокий уровень, то разрешен сброс, если низкий, то сброс неактивен.
[0]	INTEN	ЧТ/ЗП	Сигнал, разрешающий прерывание события, WDOGINT. Если установлен высокий уровень, то счетчик находится в активном состоянии и разрешены прерывания, если установлен низкий уровень, то счетчик и прерывания неактивны. Нужно перезагрузить счетчик значением, которое хранится в WdogLoad, если прерывание раннее было неактивным, но затем стало разрешенным.

WdogRIS - регистр состояния прерывания до наложения маски. Данный регистр определяет состояние прерываний, формируемых счетчиком, до наложения маски. Регистр WdogRIS возводится в единицу в момент, когда счетчик сторожевого таймера достигает нуля. В Таблица 8.46 приведен формат регистра WdogRIS.

Таблица 8.46 - Формат регистра WdogRIS

Биты	Название	Тип	Выполняемая функция
[31:1]	-	-	Зарезервировано
[0]	WDOGRIS	ЧТ	Состояние прерывания от счетчика до наложения маски.

WdogMIS - регистр состояния прерывания после наложения маски. Данный регистр определяет состояние прерывания счетчика после наложения маски. Это состояние формируется логическим функцией «И» между битом WDOGRIS и битом INTEN регистра управления. В Таблица 8.47 приведен формат регистра WdogMIS.

Таблица 8.47 - Формат регистра WdogMIS

Биты	Название	Тип	Выполняемая функция
[31:1]	-	-	Зарезервировано
[0]	WDOGMIS	ЧТ	Состояние прерывания от счетчика после наложения маски.

WdogLock - регистр блокировки записи. Запись любого значения кроме 0x1ACCE551 в данный регистр блокирует программную запись во все другие регистры блока. Запись значения 0x1ACCE551 разблокирует программную запись во все регистры блока. Таким образом, можно защитить регистры WDOG от некорректно работающего программного обеспечения.

Чтение из этого регистра возвращает состояние блокировки:

- 0 – доступ по записи разрешен (нет блокировки)
- 1 – доступ по записи запрещен (заблокирован)

В Таблица 8.48 приведен формат регистра WdogLock.

Таблица 8.48 - Формат регистра WdogLock

Биты	Название	Тип	Выполняемая функция
[31:0]	WDOGLOCK	ЧТ/ЗП	Запись значения 0x1ACCE551 в этот регистр разрешает запись во все регистры. Запись любого другого значения делает невозможным доступ по записи ко всем регистрам. Чтение возвращает состояние блокировки: 0x00000000 – запись во все регистры разрешена 0x00000001 – запись во все регистры запрещена

WdogPeriphID0-3 – Идентификатор периферийного устройства Регистр TimerPeriphID0-3 представляет четыре 8-битных регистра, предназначенных только для чтения, которые охватывают адресное пространство от 0xFE0 до0xFEC. Регистры могут быть концептуально представлены как 32-битный регистр. В Таблица 8.49 приведен формат этого регистра.

Таблица 8.49 - Формат регистра WdogPeriphID0-3

Биты	Выполняемые функции
PartNumber[11:0]	Содержит шифр компонента периферийного блока. Для DIT 0x805
DesignerID[19:12]	Идентификационный номер проектировщика блока. Для ARM 0x41 (ASCII A)
Revision[23:20]	Является номером ревизии периферийного блока. Номер ревизии начинается с нуля
Configuration[31:24]	Является вариантом конфигурации периферии. Для DIT 0x0

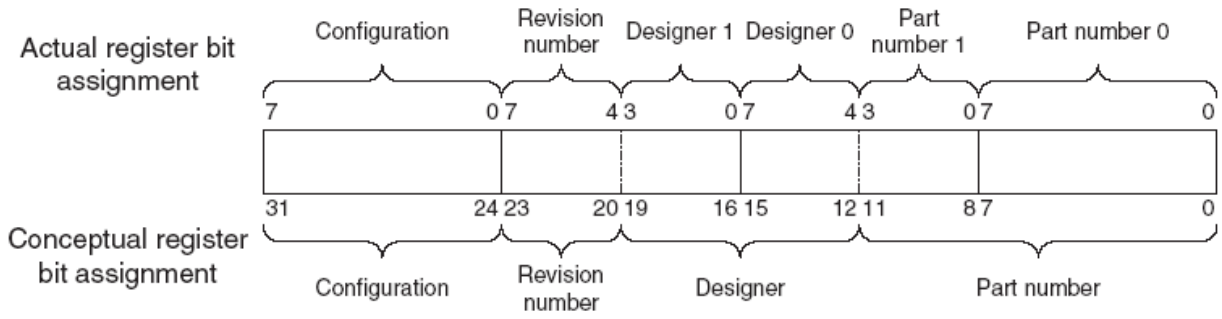


Рисунок 8.2 - Формат регистра WdogPeriphID0-3

Отметим, что при программировании важно помнить, что периферийные блоки занимают 4 Кбайт адресного пространства. Обращение к периферийным регистрам должно быть 32-битным, независимо от их реальной разрядности.

WdogPCellID0-3 – четыре 8-битных регистра, которые охватывают адресное пространство 0xFF0-0xFFC. Регистры, программно доступные только по чтению, могут быть концептуально представлены как 32-битный регистр. Регистр предназначен для идентификации стандартных периферийных блоков в составе системы. Регистр WdogPCellID установлен в 0xB105F00D. На Рисунок 8.3 приведен формат этого регистра.

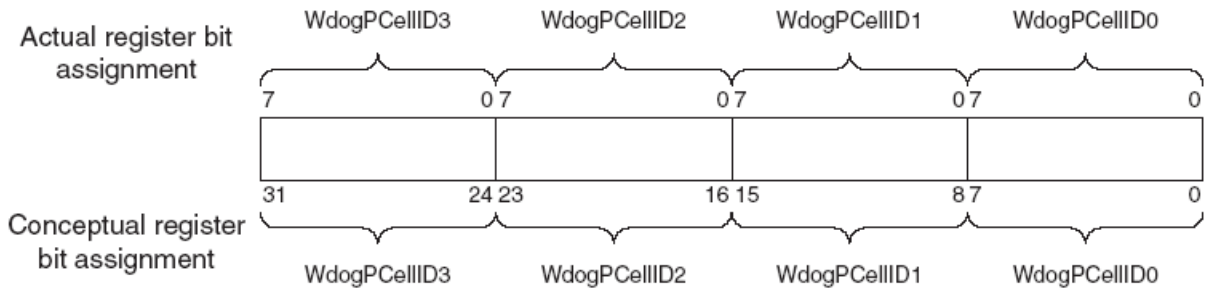


Рисунок 8.3 - Формат регистра WdogPCellID0-3

WdogITCR - регистр управления тестом интеграции. Данный однобитный регистр используется для активации тестового режима проверки правильности интеграции периферийного блока в систему – далее «теста интеграции». Когда используется этот режим, прерывание WDOGINT и сигнал сброса WDOGRST контролируются регистром WdogITOP. Формат регистра приведен в Таблица 8.50.

Таблица 8.50 - Формат регистра WdogITCR

Биты	Название	Тип	Выполняемая функция
[31:1]	-	-	Зарезервированы
[0]	ITEN	ЧТ/ЗП	Разрешение режима «теста интеграции». Когда этот бит установлен в 1, WDOG переходит в режим теста интеграции, иначе находится в нормальном режиме.

WdogITOP - регистр управления выходными сигналами. В режиме теста интеграции на выходные сигналы прерывания подаются соответствующие разряды данного регистра. Формат регистра приведен в Таблица 8.51.

Таблица 8.51 - Формат регистра WdogITOP

Биты	Название	Тип	Выполняемая функция
[31:2]	-	-	Зарезервированы, чтение не определено, должны быть записаны нули
[0]	WDOGINT	ЗП	Значение выхода WDOGINT, в режиме теста интеграции
[1]	WDOGRST	ЗП	Значение выхода WDOGRST, в режиме теста интеграции

8.9 Блок интервальных таймеров DIT

Блок интервальных таймеров содержит в своем составе два независимых 32-разрядных таймера, предназначенных для отсчета временных интервалов. Управление блоком интервальных таймеров осуществляется по шине AMBA APB версии 2.0. Частота счета счетчиков таймеров равна тактовой частоте шины APB (100 МГц). Задание режимов счета интервальных таймеров, управление выходными сигналами блока осуществляется программно.

8.9.1 Организация работы блока интервальных таймеров

Блок состоит из двух идентичных блоков Timer 1 и Timer 2, каждый из которых может функционировать в 16- или 32-битном режиме. Внутри каждого блока находится счетчик FRC (Free Running Counter). Максимальная частота счета равна тактовой частоте шины APB. Для каждого счетчика независимо можно программным образом уменьшить частоту счета в 16 или 256 раз.

Оба таймера могут быть программно настроены на работу в одном из следующих режимов:

- свободный счет (free-running) – счетчик таймера постоянно декрементируется, счет автоматически начинается с максимального значения после достижения нуля;
- периодичный (periodic) – аналогично предыдущему, только после достижения нуля счет начинается со значения, предварительно загруженного в регистр TimerXLoad;
- одиночный (one-shot) – счетчик начинает декрементироваться со значения, загруженного в регистр TimerXLoad, после достижения нуля счет останавливается.

Старт работы таймеров осуществляется по записи управляющего бита в соответствующий регистр управления. По достижении счетчиком нулевого значения вырабатывается сигнал прерывания и счетчик, в зависимости от режима работы, останавливается или начинает отсчет заново с предварительно загруженного значения. Выходы прерывания заведены на контроллер прерываний каждой процессорной системы.

8.9.2 Программно доступные регистры блока интервальных таймеров

Программно доступные регистры блока интервальных таймеров DIT расположены в области памяти периферийных устройств процессора, и имеют начальный адрес DIT Base = 0x1003_0400h и общий размер 4 Кбайт. Программно доступные регистры блока представлены в Таблица 8.52.

Таблица 8.52 - Программно доступные регистры блока DIT

Адрес	Тип	Разрядность	Начальное значение	Имя	Описание
DIT Base + 0x00	ЧТ/ЗП	32	0x00000000	Timer1Load	Регистр загрузки. TimerXLoad
DIT Base + 0x01	ЧТ	32	0xFFFFFFFF	Timer1Value	Регистр текущего значения. TimerXValue
DIT Base + 0x02	ЧТ/ЗП	8	0x20	Timer1Control	Регистр управления. TimerXControl
DIT Base + 0x03	ЗП	-	-	Timer1IntClr	Регистр сброса прерываний. TimerXIntClr
DIT Base + 0x04	ЧТ	1	0x0	Timer1RIS	Регистр состояния прерывания до наложения маски. TimerXRIS
DIT Base + 0x05	ЧТ	1	0x0	Timer1MIS	Регистр состояния прерывания после наложения маски. TimerXMIS
DIT Base + 0x06	ЧТ/ЗП	32	0x00000000	Timer1BGLoad	Фоновый регистр загрузки. TimerXBGLoad
DIT Base + 0x08	ЧТ/ЗП	32	0x00000000	Timer2Load	Регистр загрузки. TimerXLoad

Продолжение таблицы 8.52

Адрес	Тип	Разрядность	Начальное значение	Имя	Описание
DIT Base + 0x09	ЧТ	32	0xFFFFFFFF	Timer2Value	Регистр текущего значения. TimerXValue
DIT Base + 0x0A	ЧТ/ЗП	8	0x20	Timer2Control	Регистр управления. TimerXControl
DIT Base + 0x0B	ЗП	-	-	Timer2IntClr	Регистр снятия прерываний. TimerXIntClr
DIT Base + 0x0C	ЧТ	1	0x0	Timer2RIS	Регистр состояния прерывания до наложения маски. TimerXRIS
DIT Base + 0x0D	ЧТ	1	0x0	Timer2MIS	Регистр состояния прерывания после наложения маски. TimerXMIS
DIT Base + 0x0E	ЧТ/ЗП	32	0x00000000	Timer2BGLoad	Фоновый регистр загрузки. TimerXBGLoad
DIT Base + 0x0F to 0x3BF	-	-	-	-	Зарезервировано
DIT Base + 0x3C0	ЧТ/ЗП	1	0x0	TimerITCR	Регистр управления тестом интеграции. TimerITCR
DIT Base + 0x3C1	ЗП	2	0x0	TimerITOP	Регистр управления тестом интеграции. TimerITOP
DIT Base + 0x3C2 to0x3F7	-	-	-	-	Зарезервировано
DIT Base + 0x3F8	ЧТ	8	0x04	TimerPeriphID0	Идентификатор периферийного устройства. TimerPeriphID0 биты [7:0]
DIT Base + 0x3F9	ЧТ	8	0x18	TimerPeriphID1	Идентификатор периферийного устройства. TimerPeriphID1 биты [15:8]
DIT Base + 0x3FA	ЧТ	8	0x04	TimerPeriphID2	Идентификатор периферийного устройства. TimerPeriphID2 биты [23:16]
DIT Base + 0x3FB	ЧТ	8	0x00	TimerPeriphID3	Идентификатор периферийного устройства. TimerPeriphID3 биты [31:24]
DIT Base + 0x3FC	ЧТ	8	0x0D	TimerPCellID0	PrimeCell идентификатор. TimerPCellID0 биты [7:0]
DIT Base + 0x3FD	ЧТ	8	0xF0	TimerPCellID1	PrimeCell идентификатор. TimerPCellID1 биты [15:8]
DIT Base + 0x3FE	ЧТ	8	0x05	TimerPCellID2	PrimeCell идентификатор. TimerPCellID2 биты [23:16]
DIT Base + 0x3FF	ЧТ	8	0xB1	TimerPCellID3	PrimeCell идентификатор. TimerPCellID3 биты [31:24]

Использование символа X в названии регистров означает, что регистр относится либо к блоку Timer 1, либо к блоку Timer 2.

TimerXLoad - регистр загрузки начального значения счета – 32-битный регистр, доступный по записи и по чтению. При записи в данный регистр текущее значение счетчика незамедлительно меняется на значение, записанное по активному фронту тактового сигнала. При работе счетчика в периодическом режиме, при обнулении счетчика содержимое данного регистра переписывается в счетчик и счет продолжается далее. Отметим, что минимальным

значением для TimerXLoad является 1. Если значение TimerXLoad установлено в 0, то прерывания по достижению счетчиком нуля вырабатываются незамедлительно.

Значение, считанное из регистра TimerXLoad, - это всегда то значение, которое будет прописано в счетчике, как только он станет равным нулю в периодическом режиме или по старту счета при работе в одиночном режиме.

TimerXValue – регистр текущего значения - 32-битный регистр, доступный только для чтения. Данный регистр содержит текущее значение счетчика.

После процесса загрузки, когда прописывается новое значение в TimerXLoad, в регистре TimerXValue незамедлительно отражается новое загруженное значение.

Отметим, что при работе с таймером в 16-битном режиме старшие 16 бит 32-битного регистра TimerXValue не обязательно автоматически устанавливаются в нуль. В них сохраняется значение, записанное прежде в 32-битном режиме.

TimerXControl - Регистр управления таймером – 8-разрядный регистр, предназначенный для задания режимов работы и управления таймером. Регистр доступен по записи и по чтению.

Формат регистра управления приведен в Таблица 8.53.

Таблица 8.53 - Формат регистра TimerXControl

Биты	Название	Тип	Функции
[31:8]	-	-	Резервные биты, не изменяются, игнорируются при чтении
[7]	TimerEn	ЧТ/ЗП	Бит разрешения: 0 = Timer X FRC неактивен (по умолчанию) 1 = Timer X FRC активен
[6]	TimerMode	ЧТ/ЗП	Бит режима: 0 = Timer X FRC находится в режиме свободного доступа (по умолчанию) 1 = Timer X FRC находится в периодическом режиме.
[5]	IntEnable	ЧТ/ЗП	Бит разрешения прерываний: 0 = Timer X FRC Прерывание неактивно 1 = Timer X FRC Прерывание активно (по умолчанию).
[4]	-	-	Резервные биты, не изменяются, игнорируются при чтении
[3:2]	TimerPre	ЧТ/ЗП	Биты масштабирования частоты счета (делитель частоты): 00 = синхросигнал не делится (по умолчанию) 01 = синхросигнал делится на 16 10 = синхросигнал делится на 256 11 = не используется.
[1]	TimerSize	ЧТ/ЗП	Выбирает, какой (16- или 32-битный) счетчик используется: 0 = 16-битный счетчик (по умолчанию) 1 = 32-битный счетчик.
[0]	OneShot	ЧТ/ЗП	Выбирает однократный или многократный режим счетчика: 0 = многократный режим (по умолчанию) 1 = однократный режим.

Состояние счетчика, разрядность или делитель частоты – это установки, которые не должны меняться, пока таймер работает. При программировании новой конфигурации таймер должен находиться в неактивном состоянии. После изменения конфигурации таймер должен быть заново активирован.

TimerXIntClr - регистр сброса прерываний. Любая запись в данный регистр сбрасывает запрос на прерывание от таймера.

TimerXRIS - регистр состояния прерывания до наложения маски. Регистр доступен по чтению и показывает состояние прерываний таймера до наложения маски. Формат регистра представлен в Таблица 8.54.

Таблица 8.54 - Формат регистра TimerXRIS

Биты	Имя	Тип	Функция
[31:1]	-	-	Резервные биты, не изменяются, игнорируются при чтении
[0]	TimerXRIS	ЧТ	Состояние необработанных прерываний счетчика

TimerXMIS - регистр состояния прерывания после наложения маски. Данный регистр доступен для чтения и показывает состояние прерывания таймера после наложения маски. Это состояние формируется логическим функцией «И» между битом запроса на прерывание и битом разрешения прерываний (IntEnable) регистра управления и соответствует значению выходного сигнала прерывания блока. Формат регистра представлен в Таблица 8.55.

Таблица 8.55 - Формат регистра TimerXMIS

Биты	Имя	Тип	Функция
[31:1]	-	-	Резервные биты, не изменяются, игнорируются при чтении
[0]	TimerXMIS	ЧТ	Состояние бита, разрешающего прерывание от счетчика

TimerXBGLoad - фоновый регистр загрузки – 32-битный регистр, доступный по записи и по чтению. Данный регистр содержит начальное значение счета, которое загружается в таймер по достижении таймером нулевого значения при работе в периодическом режиме.

Работа с этим регистром предоставляет альтернативный метод доступа к регистру TimerXLoad. Отличием данного метода является то, что запись в TimerXBGLoad не означает, что счетчик устанавливает новое значение сразу. Если запись в регистр TimerXLoad приводит к немедленной загрузке записанного значения в таймер, то значение, записанное в регистр TimerXBGLoad, перегружается в регистр TimerXLoad только по достижении таймером нулевого значения.

Значение, считанное из регистра TimerXBGLoad, - это всегда то значение, которое будет прописано в счетчике, как только он станет равным нулю в периодическом режиме (аналогично регистру TimerXLoad.).

TimerPeriphID0-3 - регистр - идентификатор периферийного устройства представляет собой четыре 8-битных регистра, доступных только для чтения, расположенных по адресам от 0xFE0 до 0xFEC. Регистры могут быть концептуально представлены как 32-битный регистр. В Таблица 8.56 приведен формат полей этого регистра.

Таблица 8.56 - Поля идентификатора периферийного устройства TimerPeriphID0-3

Биты	Выполняемые функции
PartNumber[11:0]	Содержит шифр компонента периферийного блока. Для DIT – 0x804
DesignerID[19:12]	Идентификационный номер проектировщика блока. Для ARM – 0x41 (ASCII A)
Revision[23:20]	Является номером ревизии периферийного блока. Номер ревизии начинается с нуля
Configuration[31:24]	Является вариантом конфигурацией периферии. Для DIT 0x0

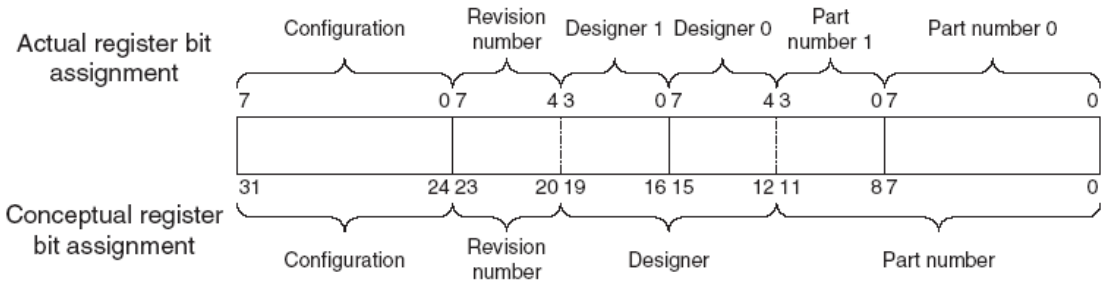


Рисунок 8.4 - Формат регистра TimerPeriphID0-3

Отметим, что при программировании важно помнить, что периферийные блоки занимают 4 Кбайт адресного пространства. Обращение к периферийным регистрам должно быть 32-битным, независимо от их реальной разрядности.

TimerPCellID0-3 – PrimeCell идентификатор - четыре 8-битных регистра, которые охватывают адресное пространство 0xFF0-0xFFC. Регистры, предназначенные только для чтения, могут быть концептуально представлены как 32-битный регистр. Регистр предназначен для идентификации стандартных периферийных блоков в составе системы. Регистр TimerPCellID установлен в 0xB105F00D. На Рисунок 8.5 приведен формат этого регистра.

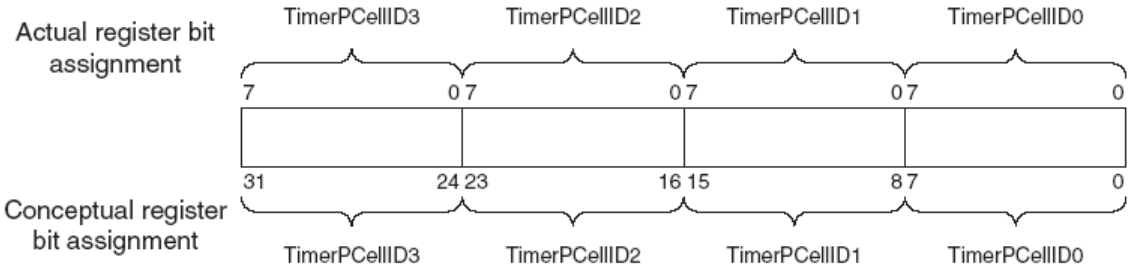


Рисунок 8.5 - Формат регистра TimerPCellID0-3

TimerITCR - регистр управления тестом интеграции. Запись в однобитный регистр используется для активации тестового режима проверки правильности интеграции периферийного блока в систему. В этом режиме выходы прерывания напрямую формируются битами регистра TimerITOP. Формат регистра приведен в Таблица 8.57.

Таблица 8.57 - Формат регистра TimerITCR

Биты	Название	Тип	Выполняемая функция
[31:1]	-	-	Зарезервированы, чтение не определено, должны быть записаны нули
[0]	ITEN	ЧТ/ЗП	Разрешение режима «теста интеграции». Когда этот бит установлен в 1, DIT переходит в режим теста интеграции, иначе находится в нормальном режиме.

TimerITOP - регистр управления выходными сигналами. Регистр доступен по чтению и по записи. В режиме теста интеграции блока выходные сигналы прерывания непосредственно формируются битами этого регистра. Комбинированное прерывание TIMERINTC формируется по логическому «ИЛИ» между битами, установленными в регистре TimerITOP. Формат регистра приведен в Таблица 8.58.

Таблица 8.58 - Формат регистра TimerITOP

Биты	Название	Тип	Выполняемая функция
[31:2]	-	-	Зарезервированы, чтение не определено, должны быть записаны нули
[0]	TIMERINT2	ЗП	Значение с выхода TIMERINT2, в режиме теста интеграции
[1]	TIMERINT1	ЗП	Значение с выхода TIMERINT1, в режиме теста интеграции

8.10 Контроллер внешних прерываний EXTIRC

Контроллер внешних прерываний предназначен для фиксации определенных событий на внешних выводах микросхемы и формирования на основе этого запросов на прерывание к процессорным системам. Контроллер имеет 4 независимых входа, изменение состояния которых приводит к формированию прерываний. Управление и настройка контроллера внешних прерываний осуществляется по шине AMBA APB версии 2.0. В Таблица 8.59 приведен список внешних выводов, относящихся к контроллеру прерываний.

Таблица 8.59 - Выводы микросхемы, подключенные к контроллеру прерываний

Вывод	Тип буфера	Примечание
INT0…INT3	I	Входы внешних прерываний 0…3

Для каждого из четырех внешних выводов можно независимо задать вид события, на основании которого контроллер будет формировать запрос на внешнее прерывание:

- Низкий уровень на входе;
- Высокий уровень на входе;
- Фронт сигнала на входе;
- Срез сигнала на входе.

Изменение состояния на каждом из входов блока может быть независимо замаскировано и не приводить к формированию запроса на прерывание.

8.10.1 Программно доступные регистры контроллера прерываний

Программно доступные регистры контроллера прерываний EXTIRC расположены в области памяти периферийных устройств процессора имеют начальный адрес EXTIRC Base = 0x10031400 и общий размер 4 Кбайт. Перечень программно доступных регистров представлен в Таблица 8.60.

Таблица 8.60 - Программно доступные регистры контроллера EXTIRC

Адрес	Тип	Разрядность	Начальное значение	Имя	Описание
EXTIRC Base +0x00	ЧТ/ЗП	4	0x0	EIENB	Регистр разрешения внешних прерываний
EXTIRC Base +0x01	ЧТ/ЗП	4	0x0	EIREQ	Регистр запроса внешних прерываний
EXTIRC Base +0x02	ЧТ/ЗП	8	0x55	EILVL	Регистр уровня внешних прерываний
EXTIRC Base + 0x03 to 0x3FF	-	-	-	-	Зарезервировано

EIENB – регистр разрешения внешних прерываний. В Таблица 8.61 приведен формат регистра EIENB.

Таблица 8.61 - Формат регистра EIENB

Биты	Название бит	Выполняемая функция
[31:4]	-	Зарезервировано
[3]	ENB3	Если бит ENB нуль, то по соответствующему входу внешнее событие маскируется. 0 – формирование прерывания по внешнему событию замаскировано (по умолчанию) 1 – формирование прерывания разрешено
[2]	ENB2	
[1]	ENB1	
[0]	ENB0	

EIREQ – регистр запросов прерываний. В Таблица 8.62 приведен формат регистра EIREQ.

Таблица 8.62 - Формат регистра EIREQ

Биты	Название бит	Выполняемая функция
[31:4]	-	Зарезервировано
[3]	REQ3	Если есть запрос на прерывание, бит REQ устанавливается в 1. 0 – Нет запроса на прерывание (по умолчанию) 1 – Есть запрос на прерывание
[2]	REQ2	
[1]	REQ1	
[0]	REQ0	

EILVL – регистр типа внешних событий. В Таблица 8.63 приведен формат регистра EILVL.

Таблица 8.63 - Формат регистра EILVL

Биты	Название бит	Выполняемая функция
[31:8]	-	Зарезервировано
[7:6]	LVL3[1:0]	Ниже приведены возможные значения бит LVLx[1:0] для каждого из четырех каналов. Значение этого регистра определяет, при каком типе внешнего прерывания будет генерироваться внутреннее прерывание. 00 – внутреннее прерывание генерируется при низком уровне внешнего сигнала 01 – внутреннее прерывание генерируется при высоком уровне внешнего сигнала (по умолчанию) 10 – внутреннее прерывание генерируется по переднему фронту внешнего сигнала 11 - внутреннее прерывание генерируется по заднему фронту внешнего сигнала
[5:4]	LVL2[1:0]	
[3:2]	LVL1[1:0]	
[1:0]	LVL0[1:0]	

8.11 Блок программируемых выводов общего назначения GPIO

Блок программируемых выводов общего назначения предназначен для приема информации о состоянии внешних устройств, подключенных к процессору, и выдачи низкочастотных сигналов управления на эти устройства. Блок имеет 16 внешних выводов, каждый из которых может быть независимо сконфигурирован как вход или как выход. Управление блоком GPIO осуществляется по шине AMBA APB версии 2.0. В Таблица 8.64 приведен список внешних выводов, относящихся к GPIO порту.

Таблица 8.64 - Выводы микросхемы, подключенные к блоку GPIO

Вывод	Тип буфера	Примечание
GPIO0…GPIO15	I/O	Программируемые порты ввода/вывода 0…15

8.11.1 Программно доступные регистры блока GPIO

Программно доступные регистры контроллера портов общего назначения GPIO расположены в области памяти периферийных процессора и имеют начальный адрес GPIO Base = 0x10030C00 и общий размер 4 Кбайт. Спецификация регистров представлена в Таблица 8.65.

Таблица 8.65 - Спецификация регистров контроллера GPIO

Адрес	Тип	Разряд-ность	Начальное значение	Имя	Описание
GPIO Base + 0x00	ЧТ/ЗП	8	-	PDR0	Регистр порта данных 0
GPIO Base + 0x01	ЧТ/ЗП	8	-	PDR1	Регистр порта данных 1
GPIO Base + 0x02 to 0x03	-	-	-	-	Резерв (доступ запрещен)
GPIO Base + 0x04	ЧТ/ЗП	8	0x00	DDR0	Регистр направления передачи данных 0
GPIO Base + 0x05	ЧТ/ЗП	8	0x00	DDR1	Регистр направления передачи данных 1
GPIO Base + 0x06 to 0x3FF	-	-	-	-	Резерв (доступ запрещен)

PDRx (x=0, 1) – два регистра, используемые для записи и чтения значений на внешних выводах GPIO. В Таблица 8.66 приведен формат регистров PDRx.

Таблица 8.66 - Формат регистров PDRx

Биты	Название	Тип	Выполняемая функция
[31:8]	-	-	Зарезервировано
[7:0]	PDRx	ЧТ/ЗП	Входные и выходные данные передаются через данный 8-битный регистр. Биты распределены по регистрам следующим образом: PDR0[7:0] - внешние выводы GPIO[7:0] PDR1[7:0] - внешние выводы GPIO[15:8] Направление передачи порта определяется соответствующими битами регистра DDR.

DDRx (x=0, 1) – два регистра, используемые для задания направления передачи данных портов GPIO. В Таблица 8.67 приведен формат регистров DDRx.

Таблица 8.67 - Формат регистров DDRx

Биты	Название	Тип	Выполняемые функции
[31:8]	-	-	Зарезервировано
[7:0]	DDR	ЧТ/ЗП	DDR регистры – регистры, контролирующие направление передачи данных портов GPIO. Каждый i-й бит DDRx регистра (i =7,…,0) управляет направлением, соответствующего ему GPIO порта. DDRx[i] = 0 - порт входных данных DDRx[i] = 1 - порт выходных данных Биты распределены по регистрам следующим образом: DDR0[7:0]: управление направлением портов GPIO[7:0]; DDR1[7:0]: управление направлением портов GPIO[15:8].

После системного сброса порты GPIO[15:0] установлены как входы.

8.12 Контроллер синхронного последовательного интерфейса SPI

Контроллер последовательного интерфейса предназначен для подключения стандартных устройств с интерфейсами Motorola SPI, Texas Instruments SPI или National Semiconductors Microwire. Контроллер обеспечивает подключение до 4 внешних устройств в режиме разделения времени. Максимальная скорость передачи данных – 50 Мбит/сек. Контроллер поддерживает режимы mode 0,1,2 и 3 спецификации интерфейса SPI. Длина SPI фрейма задается программно. Управление контроллером осуществляется по шине AMBA APB версии 2.0: Обмен данными возможен как в программном режиме, так и в режиме ПДП.

В Таблица 8.68 приведен список внешних выводов, относящихся к SPI порту.

Таблица 8.68 - Выводы микросхемы, входящие в состав SPI порта

Вывод	Тип буфера	Примечание
SPICLK	O	Синхросигнал SPI интерфейса
SPITXD	O	Выход данных SPI порта
SPIRXD	I	Вход данных SPI порта
SPI_CS0…SPI_CS3	O	Выбор ведомого SPI устройства 0…3 (сигнал с активным низким уровнем)

Выбор активного SPI slave устройства осуществляется путем предварительной установки/сброса бит [1:0] регистра SPI_CS в блоке системного контроллера (SC) в соответствии с Таблица 8.69.

Таблица 8.69 - Выбор активного ведомого SPI устройства

SPI_CS[1:0]	Примечание
00	Активное устройство подключено к SPI_CS0 (по умолчанию)
01	Активное устройство подключено к SPI_CS1
10	Активное устройство подключено к SPI_CS2
11	Активное устройство подключено к SPI_CS3

Контроллер SPI порта выполняет преобразование 16-разрядных данных, записываемых процессором или DMA контроллером через шину APB, в последовательное однобитное представление при выдаче данных, а также обратное преобразование при приеме данных. Буферизация передаваемых и принимаемых данных осуществляется с помощью двух раздельных 16-разрядных FIFO буферов глубиной 8 слов. Количество данных в этих буферах отслеживается по прерыванию или методом программного поллинга.

Частота передачи и приема данных (F_{spi}) программируется путем записи значений делителей частоты работы контроллера F_{apb}=100МГц в поля SCR регистра SSPCR0 и поле CPSDVSR регистра SSPCPSR, по следующей формуле:

$$F_{spi} = F_{apb}/[CPSDVR*(1+SCR)],$$

где CPSDVR = 2…254, а SCR = 0…255. Таким образом, возможна работа SPI интерфейса на частотах от 1,54 кГц до 50 МГц.

Если предполагается использовать SPI flash память для начальной загрузки СБИС, то подключаемая микросхема памяти должна удовлетворять следующим условиям:

- Поддержка SPI mode 3;
- Частота работы не менее 12,5 МГц, т. к. начальный делитель опорного синхросигнала, выставляемый загрузчиком, – 8;
- Поддержка команды FAST_READ (код – 0x0B);
- Загрузочная flash-память должна быть подключена к SPI_CS0 выводу СБИС.

8.12.1 Протокол передачи данных в различных режимах работы SPI интерфейса

Работа SPI интерфейса возможна в четырех режимах (mode 0,1,2 и 3), отличающихся друг от друга состоянием сигнала SPICLK в неактивном состоянии порта и активной фазой этого сигнала. Режим работы задается путем записи соответствующих значений в биты SPO и SPH регистра SSPCR0 контроллера.

Бит SPO управляет состоянием выхода SPICLK в режиме ожидания. Если SPO = 0, то вывод SPICLK имеет низкий уровень при отсутствии передачи данных. Если SPO = 1, то вывод SPICLK имеет высокий уровень при отсутствии передачи данных.

Бит SPH управляет активной фазой сигнала SPICLK. Если SPH = 0, то данные защелкиваются по первому изменению (фронту или срезу) сигнала SPICLK. Если SPH = 1, то данные защелкиваются по второму изменению (фронту или срезу) сигнала SPICLK.

SPI mode 0 (SPO=0, SPH=0)

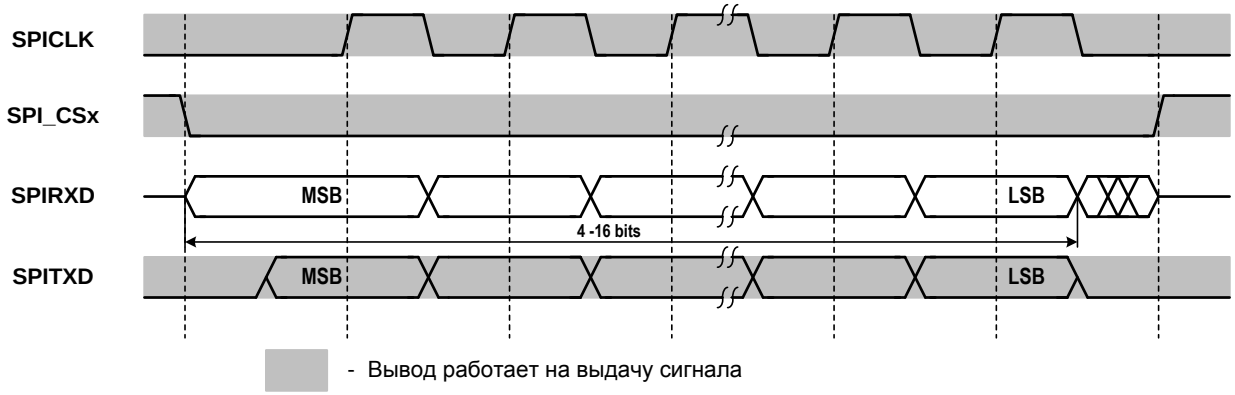


Рисунок 8.6 - SPI mode 0 (SPO=0, SPH=0) одиночная передача

Выдача данных происходит по срезу сигнала SPICLK, защелкивание входных данных – по фронту сигнала SPICLK. На Рисунок 8.7 приведены временные диаграммы передачи, состоящей из нескольких пакетов.

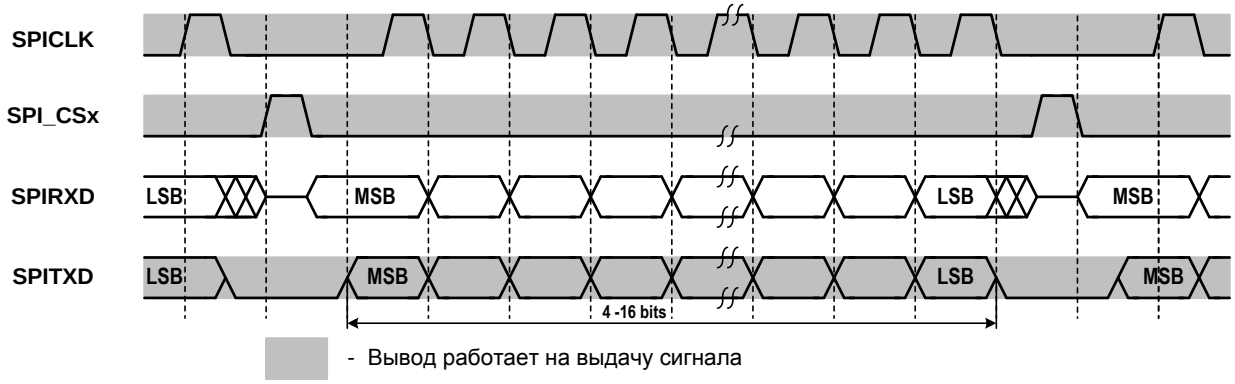


Рисунок 8.7 - SPI mode 0 (SPO=0, SPH=0) передача из нескольких пакетов

В промежутках между пакетами (словами) вывод SPI_CSx принимает значение 1 на протяжении полутакта синхросигнала SPICLK.

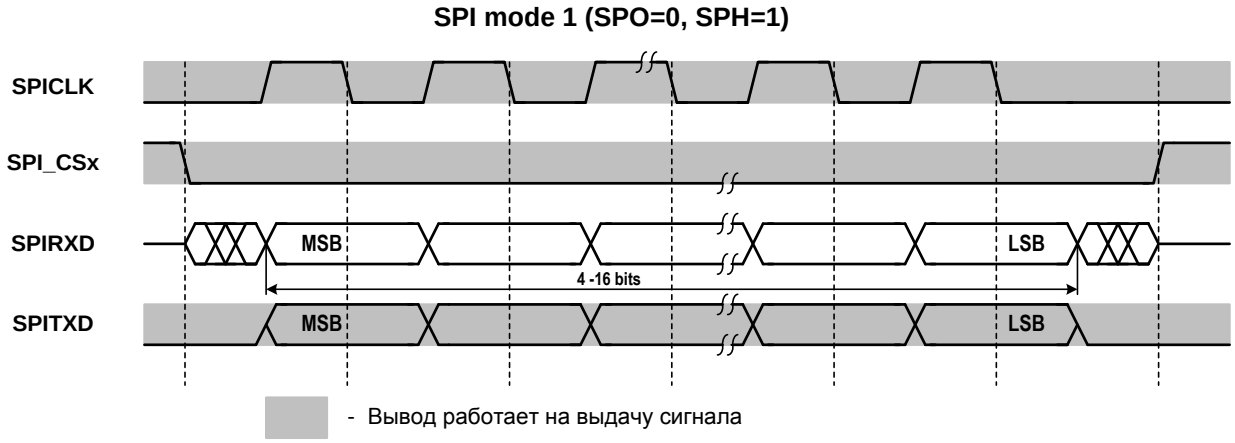


Рисунок 8.8 - SPI mode 1 (SPO=0, SPH=1) одиночная передача

Выдача данных происходит по фронту сигнала SPICLK, защелкивание входных данных – по срезу сигнала SPICLK. Сигнал SPI_CSx не меняет свое значение между словами в передаче из нескольких пакетов.

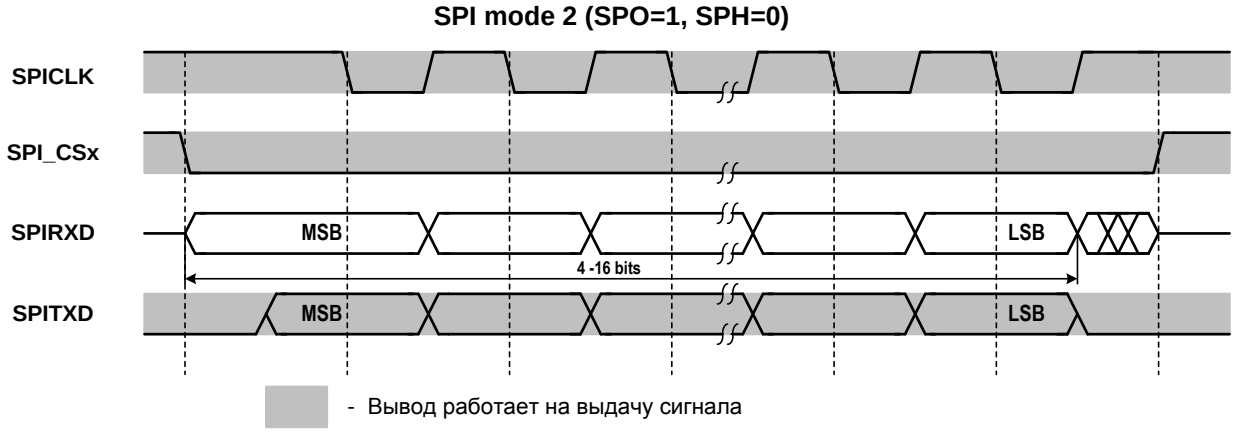


Рисунок 8.9 - SPI mode 2 (SPO=1, SPH=0) одиночная передача

Выдача данных происходит по фронту сигнала SPICLK, защелкивание входных данных – по срезу сигнала SPICLK. На Рисунок 8.10 приведены временные диаграммы передачи, состоящей из нескольких пакетов.

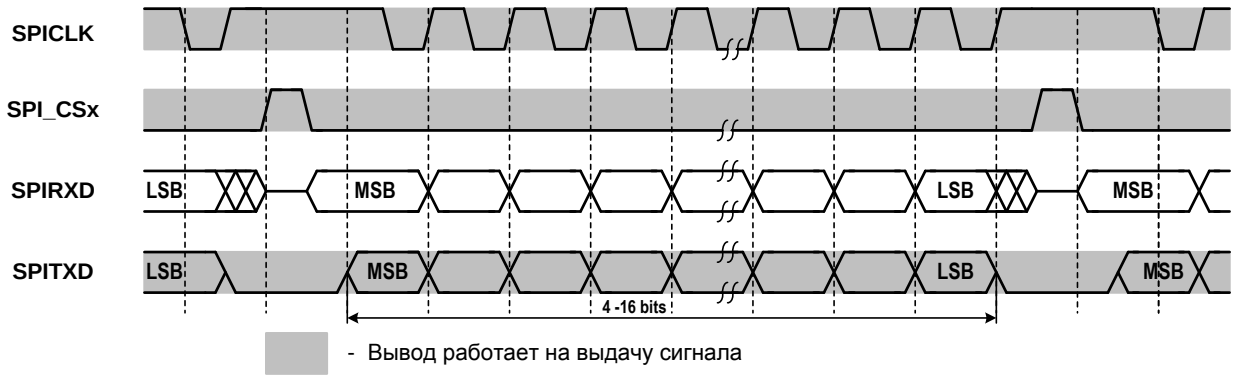


Рисунок 8.10 - SPI mode 2 (SPO=1, SPH=0) передача из нескольких пакетов

Между пакетами (словами) вывод SPI_CSx принимает значение 1 на протяжении полутакта синхросигнала SPICLK.

SPI mode 3 (SPO=1, SPH=1)

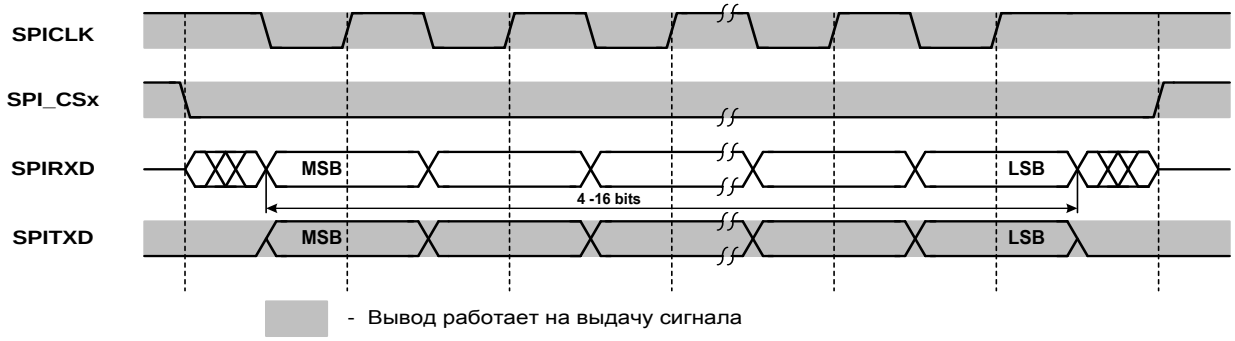


Рисунок 8.11 - SPI mode 3 (SPO=1, SPH=1) одиночная передача

Выдача данных происходит по срезу сигнала SPICLK, защелкивание входных данных по фронту сигнала SPICLK. Сигнал SPI_CSx не меняет свое значение между словами в передаче из нескольких пакетов.

8.12.2 Программно доступные регистры контроллера SPI

Программно доступные регистры контроллера порта SPI расположены в области памяти периферийных устройств процессора, имеют базовое смещение SSP Base = 0x10031000 и общий размер 4 Кбайт. Спецификация регистров представлена в Таблица 8.70.

Таблица 8.70 - Спецификация регистров контроллера порта SPI

Адрес	Тип	Разряд-ность	Начальное значение	Имя	Описание
SSP Base + 0x00	ЧТ/ЗП	16	0x0000	SSPCR0	Регистр управления 0
SSP Base + 0x01	ЧТ/ЗП	4	0x0	SSPCR1	Регистр управления 1
SSP Base + 0x02	ЧТ/ЗП	16	0x----	SSPDR	Регистр данных. SSPDR
SSP Base + 0x03	ЧТ	5	0x03	SSPSR	Регистр состояния. SSPSR
SSP Base + 0x04	ЧТ/ЗП	8	0x00	SSPCPSR	Регистр множителя синхросигнала. SSPCPSR
SSP Base + 0x05	ЧТ/ЗП	4	0x0	SSPIMSC	Регистр снятия или установки маски прерываний. SSPIMSC
SSP Base + 0x06	ЧТ	4	0x8	SSPRIS	Регистр состояния прерывания до наложения маски. SSPRIS
SSP Base + 0x07	ЧТ	4	0x0	SSPMIS	Регистр состояния прерывания после наложения маски. SSPMIS
SSP Base + 0x08	ЗП	4	0x0	SSPICR	Регистр сброса прерывания. SSPICR
SSP Base + 0x09	ЧТ/ЗП	2	0x0	SSPDMACR	Регистр управления DMA. SSPDMACR
SSP Base + 0x0A to 0x3F7	-	-	-	-	Зарезервировано
SSP base + 0x3F8	ЧТ	8	0x22	SSPPeriphID0	Идентификатор периферийного устройства. SSPPeriphID0 биты [7:0]
SSP base + 0x3F9	ЧТ	8	0x10	SSPPeriphID1	Идентификатор периферийного устройства. SSPPeriphID1 биты [15:8]
SSP base + 0x3FA	ЧТ	8	0x04	SSPPeriphID2	Идентификатор периферийного устройства. SSPPeriphID2 биты [23:16]
SSP base + 0x3FB	ЧТ	8	0x00	SSPPeriphID3	Идентификатор периферийного устройства. SSPPeriphID3 биты [31:24]
SSP base + 0x3FC	ЧТ	8	0x0D	SSPPCellID0	PrimeCell идентификатор. SSPPCellID0 биты [7:0]
SSP base + 0x3FD	ЧТ	8	0xF0	SSPPCellID1	PrimeCell идентификатор. SSPPCellID1 биты [15:8]
SSP base + 0x3FE	ЧТ	8	0x05	SSPPCellID2	PrimeCell идентификатор. SSPPCellID2 биты [23:16]
SSP base + 0x3FF	ЧТ	8	0xB1	SSPPCellID3	PrimeCell идентификатор. SSPPCellID3 биты [31:24]

SSPCR0 –регистр управления 0 контроллера. Регистр доступен по чтению и записи и определяет режим работы порта SPI, скорость передачи данных и размер слова данных. В Таблица 8.71 приведен формат регистра SSPCR0.

Таблица 8.71 - Формат регистра SSPCR0

Биты	Название	Тип	Выполняемая функция
[15:8]	SCR	ЧТ/ЗП	Serial Clock Rate. Значение SCR используется для определения скорости приема и передачи блока SSP. Скорость вычисляется по следующей формуле: $\frac{F_{SSPCLK}}{CPSDVR \times (1+SCR)}$, где CPSDVSR – значение от 2 до 254, программируемая регистром SSPCPSR, SCR может принимать значение от 0 до 255.
[7]	SPH	ЧТ/ЗП	SSPCLKOUT Phase. Фаза сигнала SSPCLKOUT
[6]	SPO	ЧТ/ЗП	SSPCLKOUT Polarity. Полярность SSPCLKOUT
[5:4]	FRF	ЧТ/ЗП	Frame Format. Формат данных: 00 – Motorola SPI формат данных 01…11 – не используется
[3:0]	DSS	ЧТ/ЗП	Data Size Select. Выбор размера данных: 0000…0010 – зарезервировано 0011 – 4-битные данные 0100 – 5-битные данные 0101 – 6-битные данные 0110 – 7-битные данные 0111 – 8-битные данные 1000 – 9-битные данные 1001 – 10-битные данные 1010 – 11-битные данные 1011 – 12-битные данные 1100 – 13-битные данные 1101 – 14-битные данные 1110 – 15-битные данные 1111 – 16-битные данные

SSPCR1 –регистр управления 1 контроллера. Регистр доступен по чтению и записи и определяет параметры работы контроллера. В Таблица 8.72 приведен формат регистра SSPCR1.

Таблица 8.72 - Формат регистра SSPCR1

Биты	Название	Тип	Выполняемая функция
[15:4]	-	-	Зарезервировано, при записи должен записываться нуль
[3]	SOD	ЧТ/ЗП	Slave-mode output disable. Бит используется только при работе SSP в режиме приема.
[2]	MS	ЧТ/ЗП	Master or Slave mode Select. Поддерживается работа только в режиме передачи (режим Master). Этот бит должен быть всегда равен 0 (значение по умолчанию).
[1]	SSE	ЧТ/ЗП	Synchronous Serial Port Enable. Управление портом: 0 – порт неактивен 1 – порт активен
[0]	LBM	ЧТ/ЗП	Loop Back Mode. Управление тестовым кольцевым режимом: 0 – нормальный режим функционирования порта 1 – выходы передающего сдвигового регистра соединены с входами принимающего сдвигового регистра

SSPDR – 16-битный регистр данных. Формат регистра представлен в Таблица 8.73. Чтение из этого регистра возвращает слово данных из приемного FIFO, на которое в данный момент указывает счетчик чтения. Данные попадают в приемное FIFO автоматически из блока принимающей логики после приема полного SPI фрейма.

Запись в SSPDR приводит к записи данных в передающее FIFO, откуда автоматически перемещаются в передающий сдвиговый регистр. Далее данные последовательно сдвигаются наружу кристалла через вывод SPITXD.

Если разрядность передаваемых данных меньше, чем 16 бит, то данные должны быть выровнены по правому краю слова. Принимаемые данные выравниваются автоматически в принимающем буфере.

Таблица 8.73 - Формат регистра SSPDR

Биты	Название	Тип	Выполняемая функция
[15:0]	DATA	ЧТ/ЗП	Если разрядность данных меньше 16 бит, то данные должны быть выровнены по правому краю слова. Старшие неиспользуемые биты игнорируются автоматически. В режиме приема происходит автоматическое выравнивание по правому краю слова.

SSPSR – регистр состояния контроллера. Регистр доступен только по чтению, и отражает текущее состояние порта SPI. В Таблица 8.74 приведен формат регистра SSPSR.

Таблица 8.74 - Формат регистра SSPSR

Биты	Название	Тип	Выполняемая функция
[15:5]	-	-	Зарезервирован, должен быть прописан нуль
[4]	BSY	ЧТ	Флаг занятости порта SPI: 0 = SPI простаивает 1 = SPI передает и/или принимает пакет данных
[3]	RFF	ЧТ	Сигнал полноты принимающего FIFO: 0 = принимающее FIFO не полное 1 = принимающее FIFO полное
[2]	RNE	ЧТ	Сигнал не пустоты принимающего FIFO: 0 = принимающее FIFO пустое 1 = принимающее FIFO не пустое
[1]	TNF	ЧТ	Сигнал неполноты передающего FIFO: 0 = передающее FIFO полное 1 = передающее FIFO не полное
[0]	TFE	ЧТ	Сигнал пустоты передающего FIFO: 0 = передающее FIFO не пустое 1 = передающее FIFO пустое

SSPCPSR- регистр множителя синхросигнала. Регистр доступен по чтению и по записи и определяет делитель, и доопределяет скорость передачи данных в соответствии с приведенной формулой. В Таблица 8.75 приведен формат регистра SSPCPSR.

Таблица 8.75 - Формат регистра SSPCPSR

Биты	Название	Тип	Выполняемая функция
[15:8]	-	-	Зарезервирован, должен быть прописан нуль
[7:0]	CPSDVSR	ЧТ/ЗП	Clock Prescale Devisor. Значение делителя синхросигнала. Должен принимать четные значения от 2 до 254, в зависимости от частоты сигнала SSPCLK. Младшие значащие биты всегда принимают значение нуль при чтении

SSPIMSC- регистр маски прерываний. Чтение регистра возвращает текущее значение маски. Запись 1 в соответствующий бит устанавливает маску, а запись 0 – снимает её. В Таблица 8.76 приведен формат регистра SSPIMSC.

Таблица 8.76 - Формат регистра SSPIMSC

Биты	Название	Тип	Выполняемая функция
[15:4]	-	-	Зарезервирован, должен быть прописан нуль
[3]	TXIM	ЧТ/ЗП	Transmit Interrupt mask. Маска прерывания передающего FIFO(SSPTXINTR): 0 = Прерывание по заполнению TxFIFO более чем наполовину замаскировано 1 = Прерывание по заполнению TxFIFO более чем наполовину не маскировано
[2]	RXIM	ЧТ/ЗП	Receive Interrupt Mask. Маска прерывания принимающего FIFO(SSPRXINTR): 0 = Прерывание по заполнению Rx FIFO менее чем наполовину замаскировано 1 = Прерывание по заполнению Rx FIFO менее чем наполовину не маскировано
[1]	RTIM	ЧТ/ЗП	Receive Timeout Interrupt Mask. Маска прерывания по превышению timeout-периода чтения из приемного FIFO (SSPRTINTR). 0 = прерывание замаскировано 1 = прерывание не маскировано
[0]	RORIM	ЧТ/ЗП	Receive Overrun Interrupt Mask. Маска прерывания при попытке записи в полное Rx FIFO: 0 = прерывание замаскировано 1 = прерывание не маскировано

SSPRIS – регистр состояния прерывания до наложения маски. Регистр доступен только по чтению и содержит текущее состояние прерываний до наложения маски. В Таблица 8.77 приведен формат регистра SSPRIS.

Таблица 8.77 - Формат регистра SSPRIS

Биты	Название	Тип	Выполняемая функция
[15:4]	-	-	Зарезервирован, должен быть прописан нуль
[3]	TXRIS	ЧТ	Определяет состояние прерывания SSPTXINTR до наложения маски
[2]	RXRIS	ЧТ	Определяет состояние прерывания SSPRXINTR до наложения маски
[1]	RTRIS	ЧТ	Определяет состояние прерывания SSPRTINTR до наложения маски
[0]	RORRIS	ЧТ	Определяет состояние прерывания SSPRORINTR до наложения маски

SSPMIS – регистр состояния прерывания после наложения маски. Регистр доступен только по чтению и содержит текущее состояние прерываний после наложения маски. В Таблица 8.78 приведен формат регистра SSPMIS.

Таблица 8.78 - Формат регистра SSPMIS

Биты	Название	Тип	Выполняемая функция
[15:4]	-	-	Зарезервирован, должен быть прописан нуль
[3]	TXMIS	ЧТ	Определяет состояние прерывания SSPTXINTR после наложения маски
[2]	RXMIS	ЧТ	Определяет состояние прерывания SSPRXINTR после наложения маски
[1]	RTMIS	ЧТ	Определяет состояние прерывания SSPRTINTR после наложения маски
[0]	RORMIS	ЧТ	Определяет состояние прерывания SSPRORINTR после наложения маски

SSPICR – регистр сброса прерывания. Регистр доступен только по записи. Запись единицы снимает соответствующее прерывание. Запись 0 не производит эффекта. В Таблица 8.79 приведен формат регистра SSPICR.

Таблица 8.79 - Формат регистра SSPICR

Биты	Название	Тип	Выполняемая функция
[15:2]	-	-	Зарезервирован, должен быть прописан нуль
[1]	RTIC	ЗП	Снимает прерывание SSPRTINTR
[0]	RORIC	ЗП	Снимает прерывание SSPRORINTR

SSPDMACR – регистр управления DMA. Регистр доступен на запись и на чтение и определяет возможность использования режима DMA на прием и на передачу. После системного сброса все разряды регистра устанавливаются в нуль. В Таблица 8.80 приведен формат регистра SSPDMACR.

Таблица 8.80 - Формат регистра SSPDMACR

Биты	Название	Тип	Выполняемая функция
[15:2]	-	-	Зарезервирован, должен быть прописан нуль
[1]	Transmit DMA Enable (TXDMAE)	ЧТ/ЗП	Если бит установлен в 1, то DMA-доступ к передающему FIFO разрешен.
[0]	Receive DMA Enable (RXDMAE)	ЧТ/ЗП	Если бит установлен в 1, то DMA-доступ к принимающему FIFO разрешен

SSPPeriphID0-3 – регистр - идентификатор периферийного устройства. Представляет четыре 8-битных регистра, предназначенных только для чтения, которые охватывают адресное пространство от 0xFE0 до 0xFEC. Регистры могут быть концептуально представлены как 32-битный регистр. Регистр предназначен для идентификации стандартных периферийных блоков в составе системы. На рисунке 8.12 приведено распределение полей, а в Таблица 8.81 приведен формат полей этого регистра.

Таблица 8.81 - Поля идентификатора периферийного устройства SS

Биты	Выполняемые функции
PartNumber[11:0]	Содержит шифр компонента периферийного блока. Для SPI (PL022) 0x022
DesignerID[19:12]	Идентификационный номер проектировщика блока. Для ARM 0x41 (ASCII A)
Revision[23:20]	Является номером ревизии периферийного блока. Номер ревизии начинается с нуля
Configuration[31:24]	Является вариантом конфигурацией периферии. Для SPI (PL022) 0x0

PPeriphID0-3

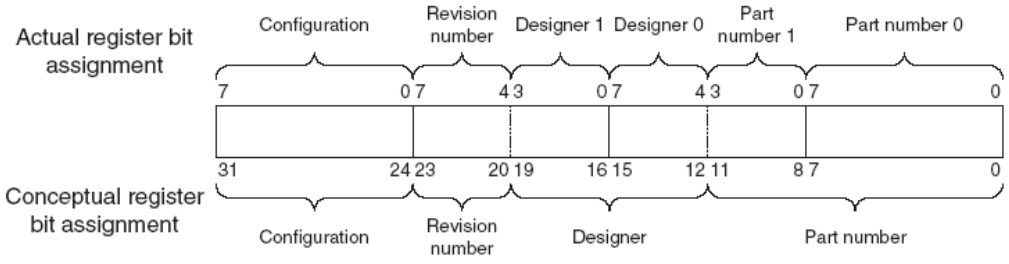


Рисунок 8.12 - Распределение полей регистра SSPPeriphID0-3

Обращение к периферийным регистрам должно быть 32-битным, независимо от их реальной разрядности.

SSPPCellID0-3 – регистр – идентификатор ШЗ блока. Регистр представляет собой четыре 8-битных регистра, которые охватывают адресное пространство 0xFF0-0xFFC. Регистры, предназначенные только для чтения, и могут быть концептуально представлены как один 32-битный регистр. Регистр предназначен для идентификации IP блока контроллера в составе системы.

Регистр SSPPCellID0-3 установлен в состояние 0xB105F00D. На Рисунок 8.13 приведено распределение полей этого регистра.

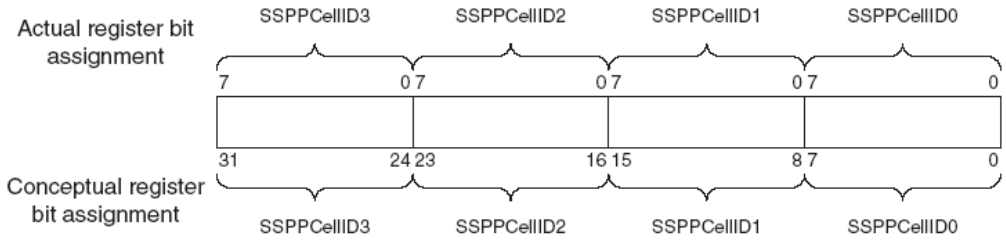


Рисунок 8.13 - Распределение полей регистра SSPPCellID0-3

8.12.3 Прерывания контроллера SPI

Контроллер формирует 5 запросов на прерывание. Четыре из них являются маскируемыми с активным высоким уровнем:

- **SSPRXINTR**. Запрос на прерывание по готовности принимаемых данных. Данный запрос устанавливается, когда принимающее FIFO содержит четыре или более слов данных;
- **SSPTXINTR**. Запрос на прерывание по готовности передающего FIFO к приему данных. Данный запрос устанавливается, когда передающее FIFO содержит менее четырех слов данных;
- **SSPRORINTR.** Запрос на прерывание по переполнению принимающего FIFO. Данный запрос устанавливается, когда приемное FIFO заполнено. Прием данных продолжается, приводя к потере данных в приемном сдвиговом регистре;
- **SSPRTINTR.** Запрос на прерывание по истечении времени ожидания приема данных (timeout). Этот запрос устанавливается, когда принятые данные находятся в приемном FIFO дольше 32 тактов. Этот механизм дополнительно оповещает программиста о том,

что необходимо забрать данные из приемного FIFO. Прерывание SSPRTINTR снимается, когда принимающее FIFO становится пустым, или если новые данные получены с SPI шины. Прерывание также может быть снято путем записи бита RTIC=1 в регистре SSPICR.

Каждый из четырех запросов на прерывание может быть замаскирован путем установления соответствующих масок в регистре SSPIMSC. Состояние запросов на прерывания до и после наложения маски можно узнать, прочитав SSPRIS и SSPMIS регистры.

Запрос на прерывание **SSPINTR** представляет собой обобщенный запрос, т. е. комбинацию всех вышеперечисленных запросов. Данный обобщенный запрос выставляется одновременно с одним из частных запросов. Обобщенный запрос подается на соответствующие входы прерываний процессорных систем.

8.13 Системный контроллер SC

Системный контроллер является вспомогательным блоком, в котором собрано управление операциями, не реализованными полностью в готовых IP-блоках. Управление и настройка системного контроллера осуществляется по шине AMBA APB версии 2.0.

Данный контроллер содержит набор программно доступных регистров, которые выполняют следующие функции:

- Управление загрузкой и запуском процессорных систем NMPU0 и NMPU1;
- Инициализация и настройка контроллера USB;
- Выбор активного устройства на шине SPI.

8.13.1 Внешние выводы системного контроллера SC

В Таблица 8.82 приведен список внешних выводов, относящихся к системному контроллеру.

Таблица 8.82 - Выводы микросхемы, относящиеся к системному контроллеру

Вывод	Тип буфера	Примечание
BOOTM[2:0]	I	Входы управления режимом начальной загрузки

Кроме этих выводов системный контроллер принимает непосредственное участие в формировании сигналов на внешних выходах SPI_CS0, SPI_CS1, SPI_CS2 и SPI_CS3 блока контроллера SPI.

8.13.2 Программно доступные регистры системного контроллера SC

Программно доступные регистры системного контроллера расположены в области памяти периферийных устройств процессора, имеют начальный адрес SC Base = 0x10030000 и общий размер 4 Кбайт. Состав программно доступных регистров представлен в Таблица 8.83.

Таблица 8.83 - Программно доступные регистры системного контроллера

Адрес	Тип	Разряд-ность	Начальное значение	Имя регистра	Описание
SC Base + 0x00	ЧТ/ЗП	32	-	NMPU_CR	Регистр управления процессорными системами NMPU0 и NMPU1
SC Base + 0x02	ЧТ/ЗП	2	b00	SPI_CS	Регистр выбора активного устройства на шине SPI
SC Base + 0x04	ЧТ/ЗП	32	h06343178	USB_CR	Регистр управления контроллером USB

NMPU_CR – регистр управления процессорными системами, используется при начальной загрузке и для запуска процессорных ядер. Формат регистра представлен в Таблица 8.84.

Таблица 8.84 - Формат регистра NMPU_CR

Биты	Название	Тип	Выполняемая функция
[31:10]	-	-	Зарезервировано
[9]	NMPU1_SIAG	ЗП	Запись значения 1 в этот бит вызывает запуск выборки инструкций в процессорной системе NMPU1. Запись значения 0 никак не влияет на устройство.
[8]	NMPU0_SIAG	ЗП	Запись значения 1 в этот бит вызывает запуск выборки инструкций в процессорной системе NMPU0. Запись значения 0 никак не влияет на устройство.
[7:3]	-	-	Зарезервировано
[2:0]	BOOTM	ЧТ	Значение на входах BOOTM[2:0]. Контроллер начальной загрузки и программа начальной инициализации СБИС используют эту информацию для определения источника начальной загрузки микросхемы. После окончания работы начального загрузчика входы BOOTM могут использоваться как входы общего назначения.

SPI_CS – регистр выбора активного устройства на шине SPI. Состояние данного регистра определяет, на какой из внешних выводов SPI_CS[3:0] будет выдан сигнал выбора внешнего устройства (Chip Select), формируемый контроллером SPI. Формат регистра приведен в Таблица 8.85.

Таблица 8.85 - Формат регистра SPI_CS

Биты	Название	Тип	Выполняемая функция
[1:0]	SPI_CS	ЧТ/ЗП	Сигнал выбора активного устройства на шине SPI выдается на вывод: 00 – SPI_CS0 01 – SPI_CS1 10 – SPI_CS2 11 – SPI_CS3

USB_CR – регистр управления контроллером USB. Регистр используется для инициализации контроллера USB и для подстройки временных параметров сигнала на шине USB. Формат регистра приведен в Таблица 8.86. Регистр работает как регистр общего назначения, то есть записанное значение хранится в нём и не может аппаратно изменяться.

Таблица 8.86 - Формат регистра USB_CR

Биты	Название	Тип	Выполняемая функция
[31:27]	-	-	Зарезервировано. Читается 0.
[26]	-	-	Зарезервировано. Читается 1.
[25:23]	COMPDISTUNE	ЧТ/ЗП	Регулировка уровня детекции отключения линии VBUS (575 мВ): • b111: +4,5% • b110: +3% • b101: +1,5% • b100: по умолчанию • b011: –1,5% • b010: –3% • b001: –4,5% • b000: –6%
[22:20]	OTGTUNE	ЧТ/ЗП	Регулировка уровня детекции подключения линии VBUS (4,65 В): • b111: +9% • b110: +6% • b101: +3% • b100: +1% • b011: по умолчанию • b010: –6% • b001: –9% • b000: –12%
[19]	-	-	Зарезервировано. Читается 0.
[18:16]	SQRXTUNE	ЧТ/ЗП	Регулировка порога обнаружения дифференциального сигнала (125 мВ): • b111: –20% • b110: –15% • b101: –10% • b100: по умолчанию • b011: +3% • b010: +5% • b001: +10% • b000: +15%
[15:12]	TXFSLSTUNE	ЧТ/ЗП	Регулировка сопротивления источника при установке высокого уровня на линиях DP и DM в режиме Full-speed (45 Ом): • b1111: –5% • b0111: –2,5% • b0011: по умолчанию • b0001: +2,5% • b0000: +5%
[11]	TXPREEMPHASISTUNE	ЧТ/ЗП	Регулировка предыскажения дифференциального сигнала: 0 – предыскажения нет (по умолчанию), 1 – предыскажение есть
[10]	TXRISETUNE	ЧТ/ЗП	Регулировка крутизны фронта и спада дифференциального сигнала: 0 – по умолчанию, 1 – фронт и спад завалены (на 8% по времени)

Продолжение таблицы 8.86

Биты	Название	Тип	Выполняемая функция
[9:6]	TXVREFTUNE	ЧТ/ЗП	Регулировка амплитуды дифференциального сигнала (400 мВ): • b1111: +8,75% • b1110: +7,5% • b1101: +6,25% • b1100: +5% • b1011: +3,75% • b1010: +2,5% • b1001: +1,25% • b1000: +1% • b0111: +0,75% • b0110: +0,5% • b0101: по умолчанию • b0100: –5% • b0011: –6,25% • b0010: –7,5% • b0001: –8,75% • b0000: –10%
[5:4]	TXHSXVTUNE	ЧТ/ЗП	Регулировка средней точки дифференциального сигнала. b11 – значение по умолчанию, b01 - -15 мВ, b10 - +15 мВ.
[3]	COMMONONN	ЧТ/ЗП	Режим отключения аналоговых блоков в режиме пониженного энергопотребления (Suspend). Значение 1 – блоки отключаются всегда, когда контроллер переходит в состояние пониженного энергопотребления (по умолчанию). Значение 0 – блоки всегда остаются включенными.
[2]	OTGDISABLE	ЧТ/ЗП	Отключение блока OTG в контроллере USB. Значение 1 – отключен, 0 – включен (по умолчанию).
[1]	PORTRESET	ЧТ/ЗП	Сброс физической части контроллера USB без перезапуска PLL. Запись значения 1 устанавливает сигнал сброса, запись значения 0 – снимает. Активное значение сигнала сброса должно быть выдержано не менее 150 мкс.
[0]	POR	ЧТ/ЗП	Общий сброс контроллера USB. Запись значения 1 устанавливает сигнал сброса, запись значения 0 – снимает. Активное значение сигнала сброса должно быть выдержано не менее 10 мкс.

8.14 JTAG интерфейс

8.14.1 Структурная схема и внешние выводы JTAG интерфейса

Микросхема имеет стандартный 5-выводной тестовый JTAG порт, реализованный согласно стандарту IEEE Std 1149.1-1990. Данный порт обеспечивает тестируемость микросхемы как на уровне самой микросхемы, так и на уровне законченного изделия. Структурная схема JTAG интерфейса представлена на Рисунок 8.14, а внешние выводы тестового порта описаны в Таблица 8.87.

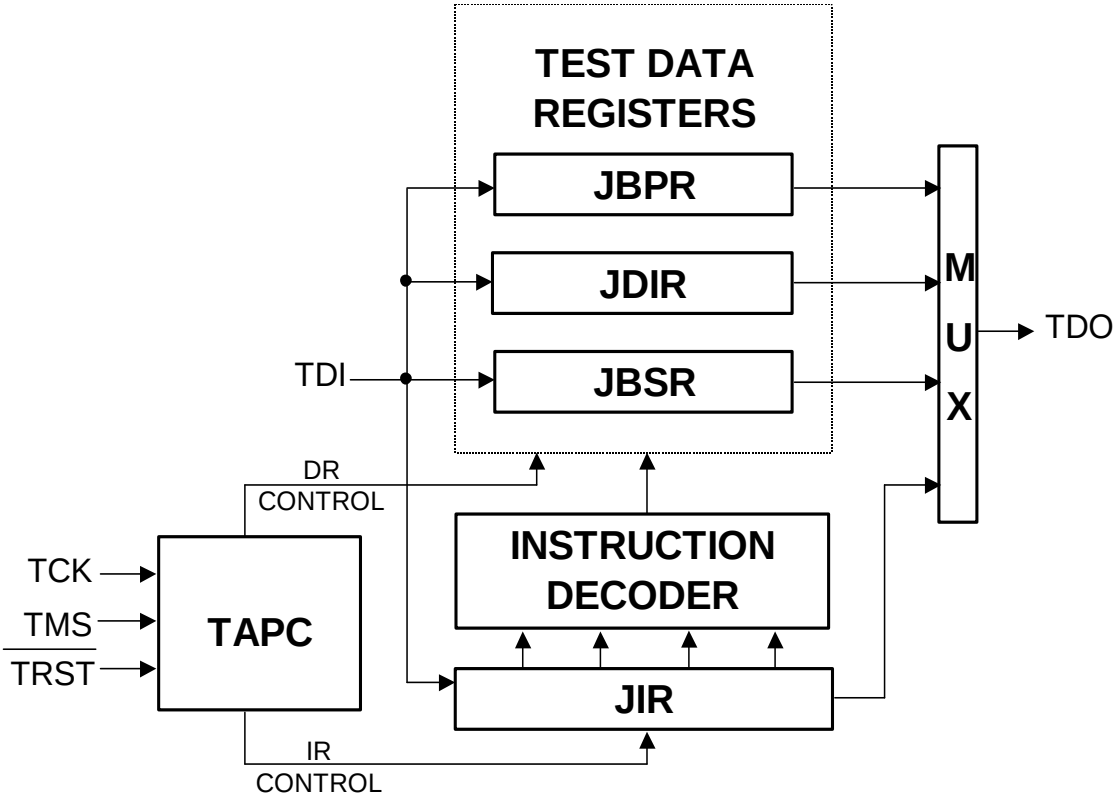


Рисунок 8.14 - Структурная схема тестового порта JTAG

Основными узлами тестового порта являются:

TAPC (Test Access Port Controller) - контроллер тестового порта. Данный контроллер реализован в виде конечного автомата, который управляет тестовой логикой в соответствии со стандартом IEEE Std 1149.1- 1990. Более подробно его работа будет описана в п. 8.14.2.

JIR (JTAG Instruction Register) - 5-разрядный регистр команд. Данный регистр является сканируемым и хранит одну из возможных команд тестового порта, более подробно он будет описан в п. 8.14.2.

JBPR (JTAG Bypass Register) - одноразрядный обходной регистр. Регистр используется для обхода пути сканирования внешних выводов, если это требуется.

JDIR (JTAG Device Identification Register) - 32-разрядный регистр идентификации устройства. Данный регистр содержит код, определяющий компанию-изготовителя, тип устройства и его версию, более подробно он будет описан в п. 8.14.3.

JBSR (JTAG Boundary Scan Register) - регистр сканирования внешних выводов. Регистр содержит одну сканируемую ячейку на каждый вход/выход и одну сканируемую ячейку на

каждый внутренний сигнал управления третьим состоянием для выходов и двунаправленных выводов, более подробно он будет описан в п. 8.14.4.

Таблица 8.87 - Внешние выводы тестового порта JTAG

Сигнал	Тип	Функциональное назначение
TDI	I	Вход тестовых данных. Данный вывод используется для ввода команд и данных при тестировании через JTAG- порт.
TDO	O	Выход тестовых данных. При вводе данных через порт TDI, они всегда последовательно выдвигаются через этот вывод. Регистры команд и данных всегда образуют сдвиговый регистр между TDI и TDO
TMS	I	Выбор режима тестирования. Данный вывод управляет состоянием конечного автомата контроллера тестового порта.
TCK	I	Тактовый сигнал тестового интерфейса
XTRST	I	Сброс тестового порта, активный низкий уровень. Данный вывод переводит контроллер тестового порта в состояние Reset. В рабочем режиме на нём должен поддерживаться низкий уровень, чтобы гарантировать правильную работу микросхемы.

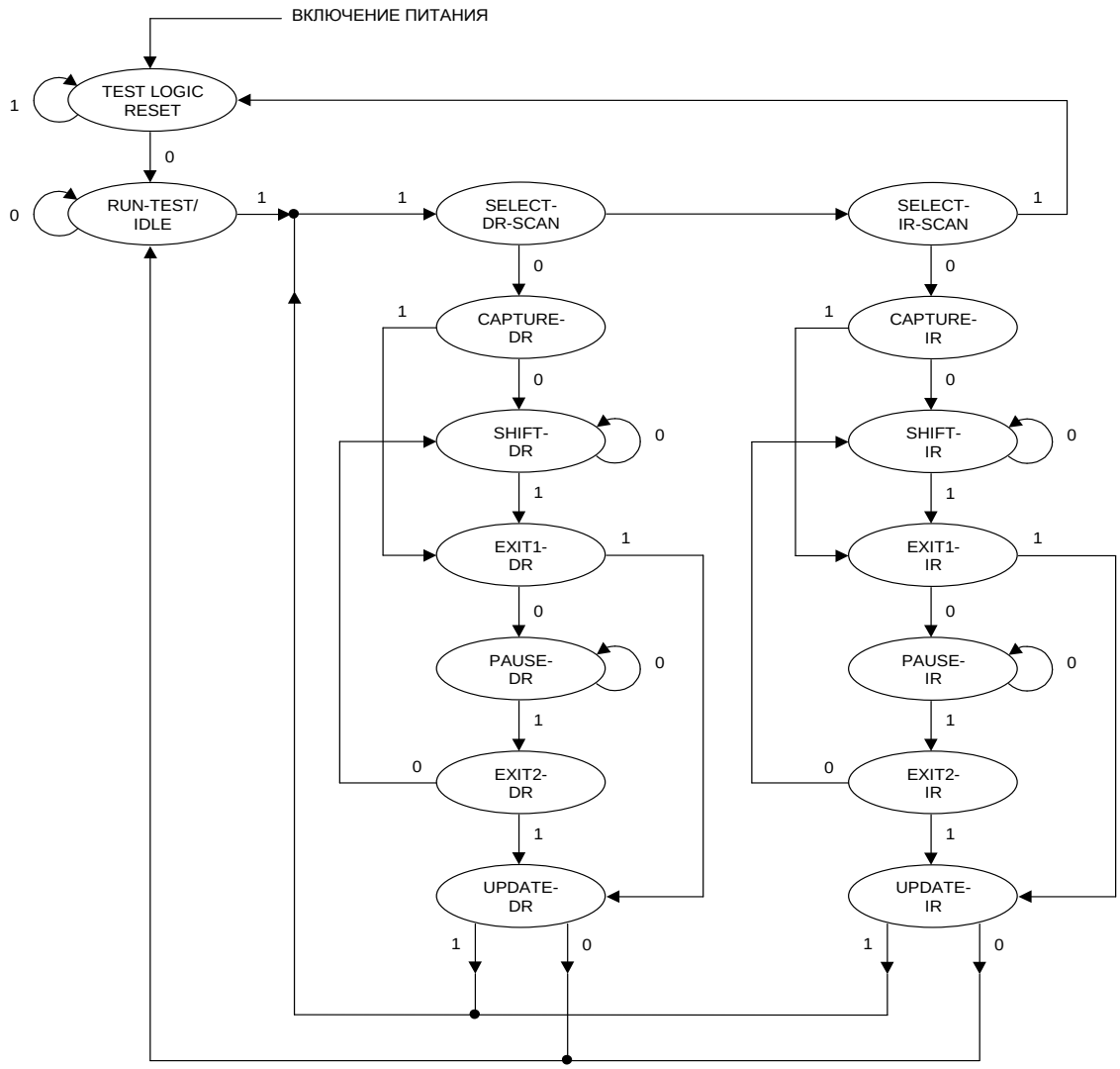
8.14.2 Управление работой тестового порта

Работа тестового порта осуществляется под управлением контроллера порта TAPC. Он представляет собой конечный автомат, который меняет свои состояния в зависимости от уровня на внешнем выводе TMS, стробируемого передним фронтом тактового сигнала TCK (см. Рисунок 8.15). Кроме того, для управления тестовым портом используются команды, последовательно загружаемые с внешнего входа TDI в регистр команд JIR. По команде выбирается то действие, которое необходимо выполнить, а также один из регистров тестовых данных - JBPR, JIDR или JBSR - для доступа к нему. Команды тестового порта приведены в Таблица 8.88.

Таблица 8.88 - Команды тестового порта

Мнемоника	Код	Функциональное описание
EXTEST	00000	Проверка связей между микросхемами на плате.
SAMPLE / PRELOAD	00001	Сканирование состояния внешних выводов.
IDCODE [1)]	00010	Выдача 32-разрядного кода идентификации устройства.
HIGHZ	11101	Перевод выводов микросхемы в высокоимпедансное состояние.
BYPASS [2)]	11111	Создание однобитового сдвигового регистра, входом которого является вывод TDI, а выходом TDO.

Примечание - Команда IDCODE всегда считывается в состоянии Capture_IR контроллера тестового порта. Команда BYPASS (11111) загружается в регистр команд в состоянии Reset контроллера тестового порта.



Состояния:

1.	Test-logic-reset	Инициализация логики тестирования
2.	Run-test/idle	Исполнение теста/режим отключения тестовой логики
3.	Shift-DR(IR)	Последовательный сдвиг в регистре данных/команд
4.	Pause- DR(IR)	Останов
5.	Select- DR(IR)	Выбор того, будут ли последующие операции производиться с регистром данных или команд
6.	Capture- DR(IR)	Режим загрузки данных с параллельных выходов регистра данных/команд
7.	Update- DR(IR)	Выдача данных на параллельные выходы регистра данных/команд (декодирование для команд)
8.	Exit- DR(IR)	Переходные состояния

Рисунок 8.15 - Диаграмма состояний контроллера тестового порта JTAG

8.14.3 Регистр идентификации устройства JDIR

JDIR – это 32-разрядный регистр, который содержит уникальный код, присвоенный микросхеме согласно стандарту IEEE Std 1149.1- 1990. Его содержимое может быть считано по команде IDCODE последовательным кодом через вывод TDO. Формат регистра идентификации устройства представлен на рисунке 8.16, а функциональное назначение его полей приведено в Таблица 8.89.

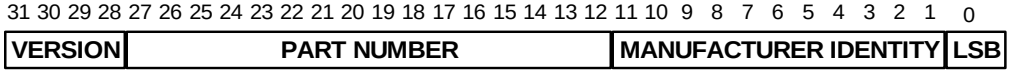


Рисунок 8.16 - Формат регистра идентификации устройства JDIR

Таблица 8.89 - Функциональное назначение полей регистра идентификации устройства JDIR

Поле	Функция	Значение
VERSION	Номер версии	0000
PART NUMBER	Код изделия	1101 1110 1010 1110
MANUFACTURER IDENTITY	Код изготовителя микросхемы	000 1000 1000
LSB	Младший бит регистра (согласно стандарту его значение всегда равно единице)	1

8.14.4 Регистр сканирования внешних выводов JBSR

JBSR – это 158-разрядный сдвиговый регистр, благодаря которому возможно сканирование внешних выводов при выполнении команды SAMPLE / PRELOAD, а также перевод их в заданное состояние при выполнении команды EXTEST. Описание каждого разряда регистра сканирования внешних выводов и выполняемой им функции приведено в таблице 8.90**.**

Таблица 8.90 - Функциональное назначение бит регистра сканирования внешних выводов JBSR

Позиция при сканировании	Обозначение	Тип	Примечание
0	C1D7	I/O	
1	XEN_0	OE	Управление выдачей C1D7
2	C1D6	I/O	
3	XEN_1	OE	Управление выдачей C1D6
4	C1D5	I/O	
5	XEN_2	OE	Управление выдачей C1D5
6	C1D4	I/O	
7	XEN_3	OE	Управление выдачей C1D4
8	C1D3	I/O	
9	XEN_4	OE	Управление выдачей C1D3
10	C1D2	I/O	
11	XEN_5	OE	Управление выдачей C1D2
12	C1D1	I/O	
13	XEN_6	OE	Управление выдачей C1D1
14	C1D0	I/O	
15	XEN_7	OE	Управление выдачей C1D0
16	XC1RDY	I/O	
17	XEN_8	OE	Управление выдачей XC1RDY
18	XC1HOLDO	O	
19	XC1STRB	I/O	
20	XEN_9	OE	Управление выдачей XC1STRB
21	XC1HOLDI	I	
22	C1IS	I	
23	U0XNMI	I	
24	C0IS	I	
25	XC0HOLDI	I	
26	XC0HOLDO	O	
27	XC0RDY	I/O	
28	XEN_10	OE	Управление выдачей XC0RDY

Продолжение таблицы 8.90

Позиция при сканировании	Обозначение	Тип	Примечание
29	XC0STRB	I/O	
30	XEN_11	OE	Управление выдачей XC0STRB
31	C0D0	I/O	
32	XEN_12	OE	Управление выдачей C0D0
33	C0D1	I/O	
34	XEN_13	OE	Управление выдачей C0D1
35	C0D2	I/O	
36	XEN_14	OE	Управление выдачей C0D2
37	C0D3	I/O	
38	XEN_15	OE	Управление выдачей C0D3
39	C0D4	I/O	
40	XEN_16	OE	Управление выдачей C0D4
41	C0D5	I/O	
42	XEN_17	OE	Управление выдачей C0D5
43	C0D6	I/O	
44	XEN_18	OE	Управление выдачей C0D6
45	C0D7	I/O	
46	XEN_19	OE	Управление выдачей C0D7
47	U0TIME0	I/O	
48	XEN_20	OE	Управление выдачей U0TIME0
49	U0TIME1	I/O	
50	XEN_21	OE	Управление выдачей U0TIME1
51	GPIO15	I/O	
52	XEN_22	OE	Управление выдачей GPIO15
53	GPIO14	I/O	
54	XEN_23	OE	Управление выдачей GPIO14
55	GPIO13	I/O	
56	XEN_24	OE	Управление выдачей GPIO13
57	GPIO12	I/O	
58	XEN_25	OE	Управление выдачей GPIO12
59	GPIO11	I/O	
60	XEN_26	OE	Управление выдачей GPIO11
61	GPIO10	I/O	
62	XEN_27	OE	Управление выдачей GPIO10
63	GPIO9	I/O	
64	XEN_28	OE	Управление выдачей GPIO9
65	GPIO8	I/O	
66	XEN_29	OE	Управление выдачей GPIO8
67	GPIO7	I/O	
68	XEN_30	OE	Управление выдачей GPIO7
69	GPIO6	I/O	
70	XEN_31	OE	Управление выдачей GPIO6
71	GPIO5	I/O	
72	XEN_32	OE	Управление выдачей GPIO5
73	GPIO4	I/O	
74	XEN_33	OE	Управление выдачей GPIO4
75	GPIO3	I/O	
76	XEN_34	OE	Управление выдачей GPIO3

Продолжение таблицы 8.90

Позиция при сканировании	Обозначение	Тип	Примечание
77	GPIO2	I/O	
78	XEN_35	OE	Управление выдачей GPIO2
79	GPIO1	I/O	
80	XEN_36	OE	Управление выдачей GPIO1
81	GPIO0	I/O	
82	XEN_37	OE	Управление выдачей GPIO0
83	WDT	I/O	
84	XEN_38	OE	Управление выдачей WDT
85	SPI_CS3	O	
86	SPI_CS2	O	
87	SPI_CS1	O	
88	SPI_CS0	O	
89	SPITXD	I/O	
90	XEN_39	OE	Управление выдачей SPITXD
91	SPIRXD	I	
92	INT0	I	
93	INT1	I	
94	INT2	I	
95	INT3	I	
96	BOOTM0	I	
97	BOOTM1	I	
98	BOOTM2	I	
99	PLLSTNDBY	I	
100	PLLPD	I	
101	PLLBP	I	
102	TM	I	
103	XRST	I	
104	USBCLKSEL	I	
105	DRVVBUS	O	
106	C3D7	I/O	
107	XEN_40	OE	Управление выдачей C3D7
108	C3D6	I/O	
109	XEN_41	OE	Управление выдачей C3D6
110	C3D5	I/O	
111	XEN_42	OE	Управление выдачей C3D5
112	C3D4	I/O	
113	XEN_43	OE	Управление выдачей C3D4
114	C3D3	I/O	
115	XEN_44	OE	Управление выдачей C3D3
116	C3D2	I/O	
117	XEN_45	OE	Управление выдачей C3D2
118	C3D1	I/O	
119	XEN_46	OE	Управление выдачей C3D1
120	C3D0	I/O	
121	XEN_47	OE	Управление выдачей C3D0
122	XC3STRB	I/O	
123	XEN_48	OE	Управление выдачей XC3STRB
124	XC3RDY	I/O	

Продолжение таблицы 8.90

Позиция при сканировании	Обозначение	Тип	Примечание
125	XEN_49	OE	Управление выдачей XC3RDY
126	XC3HOLDO	O	
127	XC3HOLDI	I	
128	C3IS	I	
129	U1XNMI	I	
130	C2IS	I	
131	XC2HOLDI	I	
132	XC2HOLDO	O	
133	XC2RDY	I/O	
134	XEN_50	OE	Управление выдачей XC2RDY
135	XC2STRB	I/O	
136	XEN_51	OE	Управление выдачей XC2STRB
137	C2D0	I/O	
138	XEN_52	OE	Управление выдачей C2D0
139	C2D1	I/O	
140	XEN_53	OE	Управление выдачей C2D1
141	C2D2	I/O	
142	XEN_54	OE	Управление выдачей C2D2
143	C2D3	I/O	
144	XEN_55	OE	Управление выдачей C2D3
145	C2D4	I/O	
146	XEN_56	OE	Управление выдачей C2D4
147	C2D5	I/O	
148	XEN_57	OE	Управление выдачей C2D5
149	C2D6	I/O	
150	XEN_58	OE	Управление выдачей C2D6
151	C2D7	I/O	
152	XEN_59	OE	Управление выдачей C2D7
153	U1TIME0	I/O	
154	XEN_60	OE	Управление выдачей U1TIME0
155	U1TIME1	I/O	
156	XEN_61	OE	Управление выдачей U1TIME1
157	IDDQ_EN	I	

Примечания

1 Каждому функциональному выводу микросхемы, исключая тестовый порт JTAG, соответствует своя ячейка регистра сканирования внешних выводов с тем же именем. Кроме того, добавляются ячейки, имеющие обозначение XEN_x, где x = 0,…,61. Они необходимы для управления двунаправленными выводами и выходами с высокоимпедансным состоянием при выполнении команды EXTEST. Если через порт JTAG в эти ячейки загружены единицы, выдача на соответствующие внешние выводы разрешена, если нули – эти выводы переводятся в высокоимпедансное состояние.

2 Используемые обозначения типов ячеек регистра сканирования внешних выводов:

- I – входная ячейка,
- – выходная ячейка,
- O(Z) - выходная ячейка с высокоимпедансным состоянием,
- I/O – входная/выходная (двунаправленная) ячейка,
- OE – ячейка управления выдачей/переводом в высокоимпедансное состояние.

9 Инициализация микросхемы

9.1 Инициализация процессора 1879ВМ6Я после включения питания

Для обеспечения корректной работы встроенных PLL необходимо сразу после включения питания подать на внешние выводы опорные тактовые сигналы PCLK, SCLK и активный (высокий) уровень на вход PLLPD. Этот уровень должен удерживаться в течение, по крайней мере, двух-трех периодов опорного тактового сигнала PCLK, но не менее 100 нс. При этом с выхода PLL тактовый сигнал не формируется, поэтому длительность высокого уровня на входе PLLPD должна быть как можно меньше. В это время происходит начальный сброс PLL и настройка счетчиков коэффициента умножения на работу на заданной частоте.

После подачи низкого уровня на вход PLLPD в течение 500 периодов опорного тактового сигнала PCLK выход PLL может быть нестабилен. Поэтому в это же время формируется внутренний сигнал системного сброса, который по логическому «ИЛИ» объединяется с внешним сигналом системного сброса (XRST).

Инициализация процессора после системного сброса описана в подразделе 9.2.

9.2 Инициализация процессора 1879ВМ6Я после системного сброса

Для корректного перевода процессора в начальное состояние, за исключением случая включения питания (см. подраздел 9.1), необходимо, чтобы сигнал на входе системного сброса (XRST) находился в активном состоянии не менее 50 тактов тактового сигнала процессора (PCLK).

Начальная инициализация может быть проведена любым ядром процессора. Выбор загружающего ядра производится в соответствии с состоянием конфигурационного вывода BOOTM2. Выбор интерфейса, по которому производится начальная инициализация, производится в соответствии с состоянием конфигурационных выводов BOOTM1, BOOTM0. Зависимость способа инициализации от состояния конфигурационных выводов BOOTM2, BOOTM1, BOOTM0 представлена в таблице 9.1.

Таблица 9.1 - Способы начальной инициализации процессора

Конфигурационный вывод			**Способ инициализации**	
BOOTM2	**BOOTM1**	**BOOTM0**	**Ядро**	**Интерфейс**
0	0	0	NMPU0	COM0
0	0	1		COM1
0	1	0		SPI
0	1	1		Резерв
1	0	0	NMPU1	COM2
1	0	1		COM3
1	1	0		SPI
1	1	1		Резерв

Состояние выводов BOOTM2, BOOTM1 и BOOTM0 фиксируется в программно доступном регистре NPU_CR блока системного контроллера SC.

Программа начального загрузчика процессора расположена во встроенном ПЗУ начальной загрузки BROM.

По сигналу системного сброса (XRST) программный счетчик каждого ядра (регистр PC) сбрасывается в состояние 0000_0000h. Аналогично адресный регистр генератора команд блоков адресных генераторов сбрасывается в состояние 0000_0000h, т. е. указывает на

нулевую ячейку нулевого банка внутренней памяти. Выборка команд при этом заблокирована.

После перевода сигнала системного сброса (XRST) в неактивное состояние запускается процедура ПДП, переписывающая содержимое BROM в нулевой банк (IMU0) внутренней памяти ядра, выбранного в качестве загружающего в соответствии с состоянием конфигурационного вывода BOOTM2. После окончания процедуры ПДП снимается блокировка выборки команд данного ядра, и программа начальной загрузки начинает выполняться.

В ходе выполнения программы начальной загрузки анализируется содержимое регистра NPU_CR блока системного контроллера SC (т. е. состояние конфигурационных выводов BOOTM1 и BOOTM0). На основании анализа выбирается интерфейс процессора, по которому будет осуществляться загрузка программы инициализации процессора, настраивается выбранный интерфейс и в программном режиме загружается программа инициализации.

Программа инициализации имеет строго заданные параметры. Независимо от того, по какому интерфейсу процессора она загружается, ее длина составляет 256 64-разрядных слов. Содержание программы определяется пользователем. Программа инициализации загружается во внутреннюю память процессорного ядра (банк IMU0) по адресу 0x0000_0000h. После окончания загрузки управление передается на загруженную программу.

В случае начальной инициализации по коммуникационным портам необходимо чтобы соответствующий порт был бы сконфигурирован в состояние “на прием”. Для этого на конфигурационный вывод CiIS необходимо подать напряжение низкого уровня.

Второе ядро процессора инициализируется первым ядром. Для этого инициализирующее ядро должно загрузить программу по начальному адресу нулевого банка памяти (IMU0) инициализируемого ядра (0x0004_0000h для NMPU0 или 0x0008_0000h для NMPU1) любым доступным ему способом и затем установить в логическую единицу бит START_IAG в регистре NPU_CR. Установка данного бита снимает блокировку адресного генератора команд инициализируемого ядра.

9.3 Рекомендации по включению процессора 1879ВМ6Я в состав вычислительной системы

При разработке аппаратуры с использованием процессора 1879ВМ6Я необходимо руководствоваться следующим:

- Выводы TM, PLLBP, PLLSTNDBY и IDDQ_EN подключить к цепи VSS;
- На выводы VREF1…VREF3 подать напряжение 0,9 В;
- Для выбора требуемых уровней лог. “0” и “1” подключить выводы C0IS, C1IS, BOOTM0, BOOTM1 и BOOTM2 к цепям VSS или VDDE соответственно;
- Вывод TXRTUNE подключить к цепи VSS через резистор номиналом 43,2 Ом ± 1%;
- Для выбора требуемых уровней лог. “0” и “1” подключить вывод USBCLKSEL к цепям VSS или VDDE соответственно;
- Если порт JTAG не используется, то вывод TDO оставить неподключенным, выводы TDI, TCK и XTRST подключить к цепи VSS, вывод TMS подключить к цепи питания буферов ввода/вывода VDDE;
- Неиспользованные выводы подключить в соответствии с рекомендациями, представленными в Таблица 10.1;

- Предусмотреть установку фильтрующих бескорпусных чип-конденсаторов емкостью 0,1 мкФ по цепям питания: по возможности, на каждый вывод питания и как можно ближе к корпусу и выводам микросхемы.
- Порядок подачи питания тактовых сигналов и сигналов системного сброса должен быть следующим:
    - Напряжения питания и тактовые сигналы должны подаваться на микросхему одновременно или последовательно в следующем порядке:
        1) Напряжение питания ядра VDDI;
        2) Напряжение питания КМОП буферов ввода/вывода VDDE;
        3) Напряжение питания SSTL буферов ввода/вывода VDDE2;
        4) Тактовые сигналы PCLK, SCLK.
    - Сигнал положительной полярности длительностью не менее 100 нс на вход PLLPD;
    - Сигнал отрицательной полярности на вход XRST длительностью не менее 50 тактов тактового сигнала процессора (PCLK).

10 Электрические, динамические и конструктивные характеристики микросхемы 1879ВМ6Я

10.1 Состав и расположение внешних выводов микросхемы

Микросхема изготавливается в 544-выводном пластиковом корпусе HSBGA (High Speed Ball Grid Array). Расположение внешних выводов процессора представлено ниже (см. Рисунок 10.1).

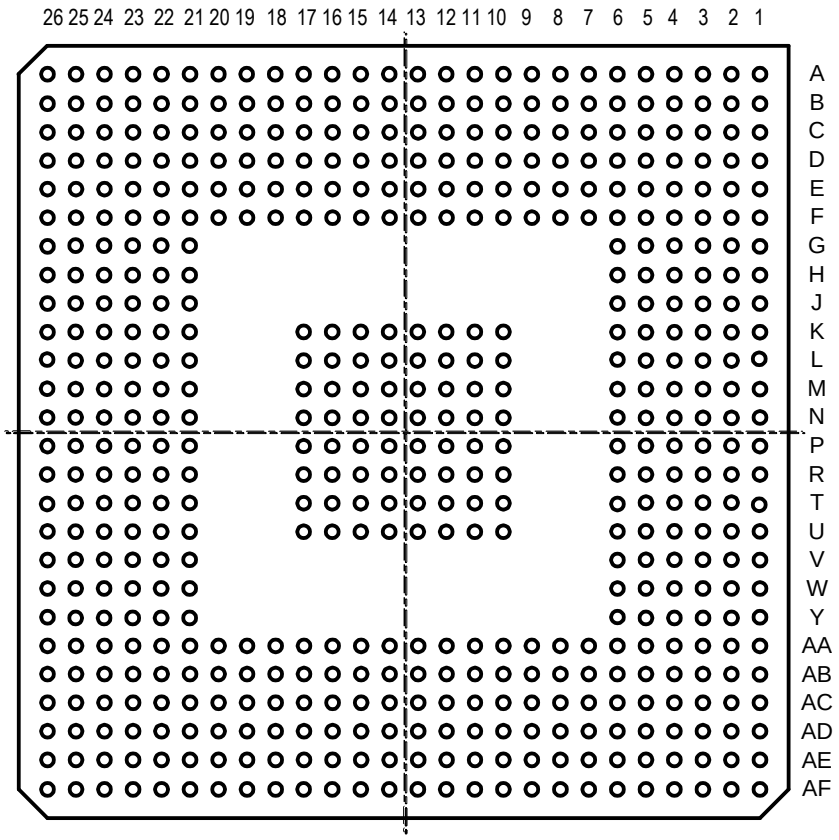


Рисунок 10.1 - Расположение внешних выводов микросхемы (вид со стороны выводов)

Таблица 10.1 приводит выводы микросхемы, сгруппированные по их функциональному назначению. В Таблица 10.2 приведены выводы микросхемы, отсортированные по соответствующим им именам сигналов в алфавитном порядке. В Таблица 10.3 приведены выводы микросхемы, отсортированные по их номерам в алфавитном порядке.

Таблица 10.1 - Выводы микросхемы в соответствии с их функциональным назначением

Обозначение [1]	Кол.	Тип [2]	Вых. ток, мА	Функциональное назначение	Выводы корпуса
1	2	3	4	5	6
Интерфейс с внешней памятью типа DDR2 (78 выводов)					
DQ0 … DQ31[3]	32	I/O		32-разрядная шина данных	E1, E2, E4, F2, G1, G6, H2, H5, J1, J3, J6, K2, L1, L6, M2, M5, AE5, AD5, AC5, AE6, AF7, AA7, AE8, AB8, AF9, AD9, AA9, AE10, AF11, AA11, AE12, AB12
A0 … A14 [4]	15	O		15 младших разрядов шины адреса	N2, N3, N4, N5, P1, P3, P4, P6, R1, R2, R5, R6, T1, T3, T6
XCS0, XCS1[4]	2	O		Выборка банков внешней памяти	W1, W3
XWE[4]	1	O		Разрешение записи в память	AA1
XRAS[4]	1	O		Строб адреса строки	Y1
XCAS[4]	1	O		Строб адреса столбца	Y2
DM0…DM3[4]	4	O		Сигналы маскирования данных при записи	F5, K5, AB6, AB10
DQS0…DQS3[3]	4	I/O		Сигналы стробов данных	G3, L3, AD7, AD11
XDQS0…XDQS3[3]	4	I/O		Инверсные сигналы стробов данных	G4, L4, AC7, AC11
BA0…BA2[4]	3	O		Сигналы выбора банка памяти	AA3, AA4, AB4
ODT0, ODT1[4]	2	O		Сигналы ODT (Memory on-die termination control signal)	W4, W6
CK0, CK1[4]	2	O		Синхросигналы шины	U2, U5
XCK0, XCK1[4]	2	O		Инверсные синхросигналы шины	V2, V5
CKE0, CKE1[4]	2	O		Сигналы разрешения синхросигналов	Y5, Y6
VREF1…VREF3[11]	3	AI		Входы напряжения смещения	J4, T4, AC9
Коммуникационный порт 0 (13 выводов)					
C0D0…C0D7[5]	8	I/O	4	Шина данных	D15, B15, E16, D16, C16, F17, E17, C17
XC0STRB[5]	1	I/O	4	Строб данных	F15
XC0RDY[5]	1	I/O	4	Готовность к приёму данных	D18
XC0HOLDI[6]	1	I		Запрос внешнего устройства на передачу данных	C14
XC0HOLDO [4]	1	O	4	Запрос процессора на передачу данных	E18
C0IS[9]	1	I		Состояние порта после системного сброса	E14

Продолжение таблицы 10.1

1	2	3	4	5	6
Коммуникационный порт 1 (13 выводов)					
C1D0…C1D7[5)]	8	I/O	4	Шина данных	D12, B12, E11, D11, C11, F10, E10, C10
XC1STRB[5)]	1	I/O	4	Строб данных	F12
XC1RDY[5)]	1	I/O	4	Готовность к приёму данных	D9
XC1HOLDI[6)]	1	I		Запрос внешнего устройства на передачу данных	C13
XC1HOLDO[4)]	1	O	4	Запрос процессора на передачу данных	E9
C1IS[9)]	1	I		Состояние порта после системного сброса	E13
Коммуникационный порт 2 (13 выводов)					
C2D0…C2D7[5)]	8	I/O	4	Шина данных	AB21, AD21, AA20, AB20, AC20, AE20, AB19, AD19
XC2STRB[5)]	1	I/O	4	Строб данных	AA18
XC2RDY[5)]	1	I/O	4	Готовность к приёму данных	AB23
XC2HOLDI[6)]	1	I		Запрос внешнего устройства на передачу данных	AA22
XC2HOLDO[4)]	1	O	4	Запрос процессора на передачу данных	AA23
C2IS[9)]	1	I		Состояние порта после системного сброса	Y22
Коммуникационный порт 3 (13 выводов)					
C3D0…C3D7[5)]	8	I/O	4	Шина данных	W24, W23, V26, V25, V23, V22, U25, U24
XC3STRB[5)]	1	I/O	4	Строб данных	W21
XC3RDY[5)]	1	I/O	4	Готовность к приёму данных	U22
XC3HOLDI[6)]	1	I		Запрос внешнего устройства на передачу данных	W26
XC3HOLDO[4)]	1	O	4	Запрос процессора на передачу данных	U21
C3IS[9)]	1	I		Состояние порта после системного сброса	Y24
Порты общего назначения (16 выводов)					
GPIO0 … GPIO15[5)]	16	I/O	4	Программируемые порты ввода/вывода	J25, H26, G25, G24, G22, G21, F26, F25, F23, F22, E24, E23, D21, E21, C20, E20
Выводы процессорных систем NMPU0 и NMPU1 (6 выводов)					
U0XNMI[6)]	1	I		Вход немаскируемого прерывания NMPU0	B13
U0TIME0[5)]	1	I/O	4	Вход/выход таймера 0 NMPU0	F19
U0TIME1[5)]	1	I/O	4	Вход/выход таймера 1 NMPU0	F20
U1XNMI[6)]	1	I		Вход немаскируемого прерывания NMPU1	Y25
U1TIME0[5)]	1	I/O	4	Вход/выход таймера 0 NMPU1	T24
U1TIME1[5)]	1	I/O	4	Вход/выход таймера 1 NMPU1	T23

Продолжение таблицы 10.1

1	2	3	4	5	6
Выводы SPI интерфейса (7 выводов)					
SPICLK[4)]	1	O(Z)	8	Синхросигнал SPI интерфейса	L21
SPITXD[4)]	1	O(Z)	4	Передаваемые данные	J22
SPIRXD[6)]	1	I		Принимаемые данные	K22
SPI_CS0… SPI_CS3[4)]	1	O(Z)	4	Выбор ведомого SPI устройства 0…3	J23, H24,
Выводы USB2.0 интерфейса full speed device (10 выводов)					
USBCLKSEL[3)]	1	I		Выбор источника тактового сигнала	M22
VBUS[4)]	1	I/O		Питание 5В USB	N24
ID[4)]	1	I		Идентификатор мини приемника USB	N26
DP[4)]	1	I/O		Линия D+ шины USB	N21
DM[4)]	1	I/O		Линия D– шины USB	P21
TXRTUNE[10)]	1	I/O		Настройка сопротивления передатчика	R22
USB_XI[7)]	1	I/O		Вывод подключения внешнего осциллятора 12,0 МГц	P23
USB_XO[4)]	1	I/O		Вывод подключения внешнего осциллятора 12,0 МГц или подключения внешнего тактового генератора 12,0 МГц	P25
USBATEST[4)]	1	I/O		Тестирование постоянной составляющей тока	R24
DRVVBUS[4)]	1	O		Управление внешней схемой генерации питания VBUS (OTG режим)	T21
JTAG интерфейс (5 выводов)					
TDO[4)]	1	O(Z)	2	Выход данных тестового порта JTAG	C8
TDI [7)]	1	I		Вход данных тестового порта JTAG	D8
TCK [7)]	1	I		Тактовый сигнал тестового порта JTAG	F7
TMS [5)]	1	I		Выбор режима тестирования JTAG	B7
XTRST[7)]	1	I		Сброс тестового порта JTAG	C7
Общее управление (18 выводов)					
INT0…INT3 [6)]	4	I		Входы внешних прерываний	K24, K25, L23, L24
BOOTM0…BOOTM2[9)]	3	I		Режим начальной загрузки процессора	L26, M23, M25
XRST [6)]	1	I		Системный сброс	AD17
PCLK_XI	1	I		Опорный тактовый сигнал процессора	AA16
PCLK_XO[4)]	1	O		Опорный тактовый сигнал процессора	AC16

Продолжение таблицы 10.1

1	2	3	4	5	6
SCLK_XI	1	I		Опорный тактовый сигнал периферии процессора	AA14
SCLK_XO[4)]	1	O		Опорный тактовый сигнал периферии процессора	AC14
PLLBP [8)]	1	I		Режим работы в обход встроенных PLL (режим “bypass”)	AB17
PLLPD	1	I		Перевод PLL в состояние “power down”	AE18
PLLSTNDBY [8)]	1	I		Перевод PLL в состояние “stand-by”	AC18
IDDQ_EN [8)]	1	I		Режим тестирования интерфейса с внешней шиной	D3
TM [8)]	1	I		Режим тестирования процессора	M26
WDT [4)]	1	O	4	Выход сторожевого таймера	J26
Питание (348 выводов)					
AVDD_PLL	4	S		Аналоговое напряжение питания блока PLL (1,0 В)	AF13, AF14, AF15, AF16
DVDD_PLL	4	S		Цифровое напряжение питания блока PLL (1,0 В)	AD13, AD14, AD15, AD16
AVDD_USB	2	S		Аналоговое напряжение питания USB (3,3 В)	P22, N25
DVDD_USB	1	S		Цифровое напряжение питания USB (1,0 В)	R25
VDDI	48	S		Напряжение питания ядра (1,0 В)	F8, F9, F11, F13, F14, F16, F18, F21, J21, K10, K11, K13, K14, K16, K17, K21, L10, L11, L13, L14, L16, L17, M21, N10, N11, N16, N17, P10, P11, P16, P17, R21, T10, T11, T13, T14, T16, T17, U10, U11, U13, U14, U16, U17, V21, Y21, AA13, AA15, AA17, AA19, AA21
VDDE	50	S		Напряжение питания буферов ввода/вывода (3,3 В)	A2, A4, A6, A8, A10, A12, A15, A17, A19, A21, A23, A25, B1, B3, B5, B9, B11, B16, B18, B20, B22, B24, B26, C2, C4, C6, C12, C15, C19, C21, C23, C25, D5, D7, D13, D14, D20, D22, D24, D26, E25, F24, G23, H25, K23, K26, N23, U23, U26, W25, Y23, AA24, AA26, AB14, AB16, AB25, AC17, AC19, AC22, AC24, AC26, AD23, AD25, AE17, AE19, AE21, AE22, AE24, AE26, AF18, AF20, AF23, AF25

Продолжение таблицы 10.1

1	2	3	4	5	6
VDDE2	42	S		Напряжение питания буферов ввода/вывода для DDR2 (1,8 В)	D1, D2, F4, F6, G2, H4, H6, J2, K4, K6, L2, M4, M6, N1, P5, R3, T5, U1, U3, V4, V6, W2, Y4, AA2, AA6, AA8, AA10, AA12, AB2, AC1, AC3, AC6, AC8, AC10, AC12, AD2, AD4, AE1, AE3, AE7, AE9, AE11, AF2, AF4
AVSS_PLL	2	S		Аналоговая земля блока PLL	AE14, AE16
DVSS_PLL	2	S		Цифровая земля блока PLL	AE13, AE15
VSSAC_USB	1	S		Аналоговая земля USB (общая часть)	R23
VSSA_USB	2	S		Аналоговая земля USB (передатчик и OTG)	N22, P24
VSS_USB	1	S		Цифровая земля USB	P26
VSS	165	S		Земля	A1,A3,A5,A7,A9,A11,A13,A14, A16,A18,A20,A22,A24,A26,B2, B4,B6,B8,B10,B14,B17,B19,B21, B23,B25,C1,C3,C5,C9,C18,C22, C24,C26,D4,D6,D10,D17,D19, D23,D25,E3,E5,E6,E7,E8,E12, E15,E19,E22,E26,F1,F3,G5,G26, H1,H3,H22,J5,J24,K1,K3,K12, K15,L5,L12,L15,L22,L25,M1, M3,M10,M11,M12,M13,M14, M15,M16,M17,M24,N6,N12, N13,N14,N15,P2,P12,P13,P14, P15,R4,R10,R11,R12,R13,R14, R15,R16,R17,R26,T2,T12,T15, T22,T25,T26,U4,U6,U12,U15, V1,V3,V24,W5,W22,Y3,Y26, AA5,AA25,AB1,AB3,AB5,AB7, AB9,AB11,AB13,AB15,AB18, AB22,AB24,AB26,AC2,AC4, AC13,AC15,AC21,AC23,AC25, AD1,AD3,AD6,AD8,AD10, AD12,AD18,AD20,AD22,AD24, AD26,AE2,AE4,AE23,AE25, AF1,AF3,AF5,AF6,AF8,AF10, AF12,AF17,AF19,AF21,AF22, AF24,AF26

Примечания

1 Для выводов со знаком инверсии активным является низкий уровень сигнала.

2 Используемые обозначения типов выводов:

I – вход, O - выход,
O(Z) - выход с высокоимпедансным состоянием, I/O – двунаправленный вывод,
AI – аналоговый вход, S – питание.

3 Каждый неиспользуемый вывод данного типа должен быть подключен к цепи VSS через резистор номиналом 10 кОм.

4 Каждый неиспользуемый вывод данного типа должен быть оставлен неподключенным.

5 Каждый неиспользуемый вывод данного типа должен быть подключен к цепи VDDE через резистор

номиналом 24 кОм.
6 Каждый неиспользуемый вывод данного типа должен быть подключен к цепи VDDE.
7 Каждый неиспользуемый вывод данного типа должен быть подключен к цепи VSS.
8 Данный вывод должен быть подключен к цепи VSS.
9 Данный вывод должен быть подключен к цепи VSS или к цепи VDDE.
10 Каждый неиспользуемый вывод данного типа должен быть подключен к цепи VSS через резистор 43,2 Ом.
11 На вывод должено быть подано напряжение 0,9 В.

Таблица 10.2 - Выводы микросхемы, отсортированные по соответствующим им именам сигналов

Сигнал	Вывод
A0	N2
A1	N3
A10	R5
A11	R6
A12	T1
A13	T3
A14	T6
A2	N4
A3	N5
A4	P1
A5	P3
A6	P4
A7	P6
A8	R1
A9	R2
AVDD_PLL	AF15
AVDD_PLL	AF16
AVDD_PLL	AF13
AVDD_PLL	AF14
AVDD_USB	P22
AVDD_USB	N25
AVSS_PLL	AE16
AVSS_PLL	AE14
BA0	AA3
BA1	AA4
BA2	AB4
BOOTM0	L26
BOOTM1	M23
BOOTM2	M25
C0D0	D15
C0D1	B15
C0D2	E16
C0D3	D16
C0D4	C16
C0D5	F17
C0D6	E17
C0D7	C17
C0IS	E14
C1D0	D12
C1D1	B12

Сигнал	Вывод
C1D2	E11
C1D3	D11
C1D4	C11
C1D5	F10
C1D6	E10
C1D7	C10
C1IS	E13
C2D0	AB21
C2D1	AD21
C2D2	AA20
C2D3	AB20
C2D4	AC20
C2D5	AE20
C2D6	AB19
C2D7	AD19
C2IS	Y22
C3D0	W24
C3D1	W23
C3D2	V26
C3D3	V25
C3D4	V23
C3D5	V22
C3D6	U25
C3D7	U24
C3IS	Y24
CK0	U2
CK1	U5
CKE0	Y5
CKE1	Y6
DM	P21
DM0	F5
DM1	K5
DM2	AB6
DM3	AB10
DP	N21
DQ0	E1
DQ1	E2
DQ10	J6
DQ11	K2
DQ12	L1

Сигнал	Вывод
DQ13	L6
DQ14	M2
DQ15	M5
DQ16	AE5
DQ17	AD5
DQ18	AC5
DQ19	AE6
DQ2	E4
DQ20	AF7
DQ21	AA7
DQ22	AE8
DQ23	AB8
DQ24	AF9
DQ25	AD9
DQ26	AA9
DQ27	AE10
DQ28	AF11
DQ29	AA11
DQ3	F2
DQ30	AE12
DQ31	AB12
DQ4	G1
DQ5	G6
DQ6	H2
DQ7	H5
DQ8	J1
DQ9	J3
DQS0	G3
DQS1	L3
DQS2	AD7
DQS3	AD11
DRVVBUS	T21
DVDD_PLL	AD15
DVDD_PLL	AD16
DVDD_PLL	AD13
DVDD_PLL	AD14
DVDD_USB	R25
DVSS_PLL	AE15
DVSS_PLL	AE13
GPIO0	J25

Продолжение таблицы 10.2

Сигнал	Вывод
GPIO1	H26
GPIO10	E24
GPIO11	E23
GPIO12	D21
GPIO13	E21
GPIO14	C20
GPIO15	E20
GPIO2	G25
GPIO3	G24
GPIO4	G22
GPIO5	G21
GPIO6	F26
GPIO7	F25
GPIO8	F23
GPIO9	F22
ID	N26
IDDQ_EN	D3
INT0	K24
INT1	K25
INT2	L23
INT3	L24
ODT0	W4
ODT1	W6
PCLK_XI	AA16
PCLK_XO	AC16
PLLBP	AB17
PLLPD	AE18
PLLSTNDBY	AC18
SCLK_XI	AA14
SCLK_XO	AC14
SPI_CS0	J23
SPI_CS1	H24
SPI_CS2	H23
SPI_CS3	H21
SPICLK	L21
SPIRXD	K22
SPITXD	J22
TCK	F7
TDI	D8
TDO	C8
TM	M26
TMS	B7
TRST	C7
TXRTUNE	R22
U0TIME0	F19
U0TIME1	F20
U0XNMI	B13

Сигнал	Вывод
U1TIME0	T24
U1TIME1	T23
U1XNMI	Y25
USB_XI	P23
USB_XO	P25
USBATEST	R24
USBCLKSEL	M22
VBUS	N24
VDDE	A2
VDDE	A4
VDDE	A6
VDDE	A8
VDDE	A10
VDDE	A12
VDDE	A15
VDDE	A17
VDDE	A19
VDDE	A21
VDDE	A23
VDDE	A25
VDDE	B1
VDDE	B3
VDDE	B5
VDDE	B9
VDDE	B11
VDDE	B16
VDDE	B18
VDDE	B20
VDDE	B22
VDDE	B24
VDDE	B26
VDDE	C2
VDDE	C4
VDDE	C6
VDDE	C12
VDDE	C15
VDDE	C19
VDDE	C21
VDDE	C23
VDDE	C25
VDDE	D5
VDDE	D7
VDDE	D13
VDDE	D14
VDDE	D20
VDDE	D22
VDDE	D24

Сигнал	Вывод
VDDE	D26
VDDE	E25
VDDE	F24
VDDE	G23
VDDE	H25
VDDE	K23
VDDE	K26
VDDE	N23
VDDE	U23
VDDE	U26
VDDE	W25
VDDE	Y23
VDDE	AA24
VDDE	AA26
VDDE	AB14
VDDE	AB16
VDDE	AB25
VDDE	AC17
VDDE	AC19
VDDE	AC22
VDDE	AC24
VDDE	AC26
VDDE	AD23
VDDE	AD25
VDDE	AE17
VDDE	AE19
VDDE	AE21
VDDE	AE22
VDDE	AE24
VDDE	AE26
VDDE	AF18
VDDE	AF20
VDDE	AF23
VDDE	AF25
VDDE2	D1
VDDE2	D2
VDDE2	F4
VDDE2	F6
VDDE2	G2
VDDE2	H4
VDDE2	H6
VDDE2	J2
VDDE2	K4
VDDE2	K6
VDDE2	L2
VDDE2	M4
VDDE2	M6

Продолжение таблицы 10.2

Сигнал	Вывод
VDDE2	N1
VDDE2	P5
VDDE2	R3
VDDE2	T5
VDDE2	U1
VDDE2	U3
VDDE2	V4
VDDE2	V6
VDDE2	W2
VDDE2	Y4
VDDE2	AA2
VDDE2	AA6
VDDE2	AA8
VDDE2	AA10
VDDE2	AA12
VDDE2	AB2
VDDE2	AC1
VDDE2	AC3
VDDE2	AC6
VDDE2	AC8
VDDE2	AC10
VDDE2	AC12
VDDE2	AD2
VDDE2	AD4
VDDE2	AE1
VDDE2	AE3
VDDE2	AE7
VDDE2	AE9
VDDE2	AE11
VDDE2	AF2
VDDE2	AF4
VDDI	F8
VDDI	F9
VDDI	F11
VDDI	F13
VDDI	F14
VDDI	F16
VDDI	F18
VDDI	F21
VDDI	J21
VDDI	K10
VDDI	K11
VDDI	K13
VDDI	K14
VDDI	K16
VDDI	K17
VDDI	K21

Сигнал	Вывод
VDDI	L10
VDDI	L11
VDDI	L13
VDDI	L14
VDDI	L16
VDDI	L17
VDDI	M21
VDDI	N10
VDDI	N11
VDDI	N16
VDDI	N17
VDDI	P10
VDDI	P11
VDDI	P16
VDDI	P17
VDDI	R21
VDDI	T10
VDDI	T11
VDDI	T13
VDDI	T14
VDDI	T16
VDDI	T17
VDDI	U10
VDDI	U11
VDDI	U13
VDDI	U14
VDDI	U16
VDDI	U17
VDDI	V21
VDDI	Y21
VDDI	AA13
VDDI	AA15
VDDI	AA17
VDDI	AA19
VDDI	AA21
Vref	J4
Vref	T4
Vref	AC9
VSS	A1
VSS	A3
VSS	A5
VSS	A7
VSS	A9
VSS	A11
VSS	A13
VSS	A14
VSS	A16

Сигнал	Вывод
VSS	A18
VSS	A20
VSS	A22
VSS	A24
VSS	A26
VSS	B2
VSS	B4
VSS	B6
VSS	B8
VSS	B10
VSS	B14
VSS	B17
VSS	B19
VSS	B21
VSS	B23
VSS	B25
VSS	C1
VSS	C3
VSS	C5
VSS	C9
VSS	C18
VSS	C22
VSS	C24
VSS	C26
VSS	D4
VSS	D6
VSS	D10
VSS	D17
VSS	D19
VSS	D23
VSS	D25
VSS	E3
VSS	E5
VSS	E6
VSS	E7
VSS	E8
VSS	E12
VSS	E15
VSS	E19
VSS	E22
VSS	E26
VSS	F1
VSS	F3
VSS	G5
VSS	G26
VSS	H1
VSS	H3

Продолжение таблицы 10.2

Сигнал	Вывод
VSS	H22
VSS	J5
VSS	J24
VSS	K1
VSS	K3
VSS	K12
VSS	K15
VSS	L5
VSS	L12
VSS	L15
VSS	L22
VSS	L25
VSS	M1
VSS	M3
VSS	M10
VSS	M11
VSS	M12
VSS	M13
VSS	M14
VSS	M15
VSS	M16
VSS	M17
VSS	M24
VSS	N6
VSS	N12
VSS	N13
VSS	N14
VSS	N15
VSS	P2
VSS	P12
VSS	P13
VSS	P14
VSS	P15
VSS	R4
VSS	R10
VSS	R11
VSS	R12
VSS	R13
VSS	R14
VSS	R15
VSS	R16
VSS	R17
VSS	R26
VSS	T2
VSS	T12
VSS	T15
VSS	T22
VSS	T25

Сигнал	Вывод
VSS	T26
VSS	U4
VSS	U6
VSS	U12
VSS	U15
VSS	V1
VSS	V3
VSS	V24
VSS	W5
VSS	W22
VSS	Y3
VSS	Y26
VSS	AA5
VSS	AA25
VSS	AB1
VSS	AB3
VSS	AB5
VSS	AB7
VSS	AB9
VSS	AB11
VSS	AB13
VSS	AB15
VSS	AB18
VSS	AB22
VSS	AB24
VSS	AB26
VSS	AC2
VSS	AC4
VSS	AC13
VSS	AC15
VSS	AC21
VSS	AC23
VSS	AC25
VSS	AD1
VSS	AD3
VSS	AD6
VSS	AD8
VSS	AD10
VSS	AD12
VSS	AD18
VSS	AD20
VSS	AD22
VSS	AD24
VSS	AD26
VSS	AE2
VSS	AE4
VSS	AE23
VSS	AE25

Сигнал	Вывод
VSS	AF1
VSS	AF3
VSS	AF5
VSS	AF6
VSS	AF8
VSS	AF10
VSS	AF12
VSS	AF17
VSS	AF19
VSS	AF21
VSS	AF22
VSS	AF24
VSS	AF26
VSS_USB	P26
VSSA_USB	P24
VSSA_USB	N22
VSSAC_USB	R23
WDT	J26
XC0HOLDI	C14
XC0HOLDO	E18
XC0RDY	D18
XC0STRB	F15
XC1HOLDI	C13
XC1HOLDO	E9
XC1RDY	D9
XC1STRB	F12
XC2HOLDI	AA22
XC2HOLDO	AA23
XC2RDY	AB23
XC2STRB	AA18
XC3HOLDI	W26
XC3HOLDO	U21
XC3RDY	U22
XC3STRB	W21
XCAS	Y2
XCK0	V2
XCK1	V5
XCS0	W1
XCS1	W3
XDQS0	G4
XDQS1	L4
XDQS2	AC7
XDQS3	AC11
XRAS	Y1
XRST	AD17
XWE	AA1

Таблица 10.3 - Выводы микросхемы, отсортированные по их номерам

Вывод	Сигнал
A1	VSS
A10	VDDE
A11	VSS
A12	VDDE
A13	VSS
A14	VSS
A15	VDDE
A16	VSS
A17	VDDE
A18	VSS
A19	VDDE
A2	VDDE
A20	VSS
A21	VDDE
A22	VSS
A23	VDDE
A24	VSS
A25	VDDE
A26	VSS
A3	VSS
A4	VDDE
A5	VSS
A6	VDDE
A7	VSS
A8	VDDE
A9	VSS
AA1	XWE
AA10	VDDE2
AA11	DQ29
AA12	VDDE2
AA13	VDDI
AA14	SCLK_XI
AA15	VDDI
AA16	PCLK_XI
AA17	VDDI
AA18	XC2STRB
AA19	VDDI
AA2	VDDE2
AA20	C2D2
AA21	VDDI
AA22	XC2HOLDI
AA23	XC2HOLDO

Вывод	Сигнал
AA24	VDDE
AA25	VSS
AA26	VDDE
AA3	BA0
AA4	BA1
AA5	VSS
AA6	VDDE2
AA7	DQ21
AA8	VDDE2
AA9	DQ26
AB1	VSS
AB10	DM3
AB11	VSS
AB12	DQ31
AB13	VSS
AB14	VDDE
AB15	VSS
AB16	VDDE
AB17	PLLBP
AB18	VSS
AB19	C2D6
AB2	VDDE2
AB20	C2D3
AB21	C2D0
AB22	VSS
AB23	XC2RDY
AB24	VSS
AB25	VDDE
AB26	VSS
AB3	VSS
AB4	BA2
AB5	VSS
AB6	DM2
AB7	VSS
AB8	DQ23
AB9	VSS
AC1	VDDE2
AC10	VDDE2
AC11	XDQS3
AC12	VDDE2
AC13	VSS
AC14	SCLK_XO

Вывод	Сигнал
AC15	VSS
AC16	PCLK_XO
AC17	VDDE
AC18	PLLSTNDBY
AC19	VDDE
AC2	VSS
AC20	C2D4
AC21	VSS
AC22	VDDE
AC23	VSS
AC24	VDDE
AC25	VSS
AC26	VDDE
AC3	VDDE2
AC4	VSS
AC5	DQ18
AC6	VDDE2
AC7	XDQS2
AC8	VDDE2
AC9	Vref
AD1	VSS
AD10	VSS
AD11	DQS3
AD12	VSS
AD13	DVDD_PLL
AD14	DVDD_PLL
AD15	DVDD_PLL
AD16	DVDD_PLL
AD17	XRST
AD18	VSS
AD19	C2D7
AD2	VDDE2
AD20	VSS
AD21	C2D1
AD22	VSS
AD23	VDDE
AD24	VSS
AD25	VDDE
AD26	VSS
AD3	VSS
AD4	VDDE2
AD5	DQ17

Продолжение таблицы 10.3

Вывод	Сигнал
AD6	VSS
AD7	DQS2
AD8	VSS
AD9	DQ25
AE1	VDDE2
AE10	DQ27
AE11	VDDE2
AE12	DQ30
AE13	DVSS_PLL
AE14	AVSS_PLL
AE15	DVSS_PLL
AE16	AVSS_PLL
AE17	VDDE
AE18	PLLPD
AE19	VDDE
AE2	VSS
AE20	C2D5
AE21	VDDE
AE22	VDDE
AE23	VSS
AE24	VDDE
AE25	VSS
AE26	VDDE
AE3	VDDE2
AE4	VSS
AE5	DQ16
AE6	DQ19
AE7	VDDE2
AE8	DQ22
AE9	VDDE2
AF1	VSS
AF10	VSS
AF11	DQ28
AF12	VSS
AF13	AVDD_PLL
AF14	AVDD_PLL
AF15	AVDD_PLL
AF16	AVDD_PLL
AF17	VSS
AF18	VDDE
AF19	VSS
AF2	VDDE2
AF20	VDDE
AF21	VSS
AF22	VSS
AF23	VDDE
AF24	VSS
AF25	VDDE

Вывод	Сигнал
AF26	VSS
AF3	VSS
AF4	VDDE2
AF5	VSS
AF6	VSS
AF7	DQ20
AF8	VSS
AF9	DQ24
B1	VDDE
B10	VSS
B11	VDDE
B12	C1D1
B13	U0XNMI
B14	VSS
B15	C0D1
B16	VDDE
B17	VSS
B18	VDDE
B19	VSS
B2	VSS
B20	VDDE
B21	VSS
B22	VDDE
B23	VSS
B24	VDDE
B25	VSS
B26	VDDE
B3	VDDE
B4	VSS
B5	VDDE
B6	VSS
B7	TMS
B8	VSS
B9	VDDE
C1	VSS
C10	C1D7
C11	C1D4
C12	VDDE
C13	XC1HOLDI
C14	XC0HOLDI
C15	VDDE
C16	C0D4
C17	C0D7
C18	VSS
C19	VDDE
C2	VDDE
C20	GPIO14
C21	VDDE

Вывод	Сигнал
C22	VSS
C23	VDDE
C24	VSS
C25	VDDE
C26	VSS
C3	VSS
C4	VDDE
C5	VSS
C6	VDDE
C7	TRST
C8	TDO
C9	VSS
D1	VDDE2
D10	VSS
D11	C1D3
D12	C1D0
D13	VDDE
D14	VDDE
D15	C0D0
D16	C0D3
D17	VSS
D18	XC0RDY
D19	VSS
D2	VDDE2
D20	VDDE
D21	GPIO12
D22	VDDE
D23	VSS
D24	VDDE
D25	VSS
D26	VDDE
D3	IDDQ_EN
D4	VSS
D5	VDDE
D6	VSS
D7	VDDE
D8	TDI
D9	XC1RDY
E1	DQ0
E10	C1D6
E11	C1D2
E12	VSS
E13	C1IS
E14	C0IS
E15	VSS
E16	C0D2
E17	C0D6
E18	XC0HOLDO

Продолжение таблицы 10.3

Вывод	Сигнал
E19	VSS
E2	DQ1
E20	GPIO15
E21	GPIO13
E22	VSS
E23	GPIO11
E24	GPIO10
E25	VDDE
E26	VSS
E3	VSS
E4	DQ2
E5	VSS
E6	VSS
E7	VSS
E8	VSS
E9	XC1HOLDO
F1	VSS
F10	C1D5
F11	VDDI
F12	XC1STRB
F13	VDDI
F14	VDDI
F15	XC0STRB
F16	VDDI
F17	C0D5
F18	VDDI
F19	U0TIME0
F2	DQ3
F20	U0TIME1
F21	VDDI
F22	GPIO9
F23	GPIO8
F24	VDDE
F25	GPIO7
F26	GPIO6
F3	VSS
F4	VDDE2
F5	DM0
F6	VDDE2
F7	TCK
F8	VDDI
F9	VDDI
G1	DQ4
G2	VDDE2
G21	GPIO5
G22	GPIO4
G23	VDDE
G24	GPIO3

Вывод	Сигнал
G25	GPIO2
G26	VSS
G3	DQS0
G4	XDQS0
G5	VSS
G6	DQ5
H1	VSS
H2	DQ6
H21	SPI_CS3
H22	VSS
H23	SPI_CS2
H24	SPI_CS1
H25	VDDE
H26	GPIO1
H3	VSS
H4	VDDE2
H5	DQ7
H6	VDDE2
J1	DQ8
J2	VDDE2
J21	VDDI
J22	SPITXD
J23	SPI_CS0
J24	VSS
J25	GPIO0
J26	WDT
J3	DQ9
J4	Vref
J5	VSS
J6	DQ10
K1	VSS
K10	VDDI
K11	VDDI
K12	VSS
K13	VDDI
K14	VDDI
K15	VSS
K16	VDDI
K17	VDDI
K2	DQ11
K21	VDDI
K22	SPIRXD
K23	VDDE
K24	INT0
K25	INT1
K26	VDDE
K3	VSS
K4	VDDE2

Вывод	Сигнал
K5	DM1
K6	VDDE2
L1	DQ12
L10	VDDI
L11	VDDI
L12	VSS
L13	VDDI
L14	VDDI
L15	VSS
L16	VDDI
L17	VDDI
L2	VDDE2
L21	SPICLK
L22	VSS
L23	INT2
L24	INT3
L25	VSS
L26	BOOTM0
L3	DQS1
L4	XDQS1
L5	VSS
L6	DQ13
M1	VSS
M10	VSS
M11	VSS
M12	VSS
M13	VSS
M14	VSS
M15	VSS
M16	VSS
M17	VSS
M2	DQ14
M21	VDDI
M22	USBCLKSEL
M23	BOOTM1
M24	VSS
M25	BOOTM2
M26	TM
M3	VSS
M4	VDDE2
M5	DQ15
M6	VDDE2
N1	VDDE2
N10	VDDI
N11	VDDI
N12	VSS
N13	VSS
N14	VSS

Продолжение таблицы 10.3

Вывод	Сигнал
N15	VSS
N16	VDDI
N17	VDDI
N2	A0
N21	DP
N22	VSSA_USB
N23	VDDE
N24	VBUS
N25	AVDD_USB
N26	ID
N3	A1
N4	A2
N5	A3
N6	VSS
P1	A4
P10	VDDI
P11	VDDI
P12	VSS
P13	VSS
P14	VSS
P15	VSS
P16	VDDI
P17	VDDI
P2	VSS
P21	DM
P22	AVDD_USB
P23	USB_XI
P24	VSSA_USB
P25	USB_XO
P26	VSS_USB
P3	A5
P4	A6
P5	VDDE2
P6	A7
R1	A8
R10	VSS
R11	VSS
R12	VSS
R13	VSS
R14	VSS
R15	VSS
R16	VSS
R17	VSS
R2	A9
R21	VDDI
R22	TXRTUNE
R23	VSSAC_USB
R24	USBATEST

Вывод	Сигнал
R25	DVDD_USB
R26	VSS
R3	VDDE2
R4	VSS
R5	A10
R6	A11
T1	A12
T10	VDDI
T11	VDDI
T12	VSS
T13	VDDI
T14	VDDI
T15	VSS
T16	VDDI
T17	VDDI
T2	VSS
T21	DRVVBUS
T22	VSS
T23	U1TIME1
T24	U1TIME0
T25	VSS
T26	VSS
T3	A13
T4	Vref
T5	VDDE2
T6	A14
U1	VDDE2
U10	VDDI
U11	VDDI
U12	VSS
U13	VDDI
U14	VDDI
U15	VSS
U16	VDDI
U17	VDDI
U2	CK0
U21	XC3HOLDO
U22	XC3RDY
U23	VDDE
U24	C3D7
U25	C3D6
U26	VDDE
U3	VDDE2
U4	VSS
U5	CK1
U6	VSS
V1	VSS
V2	XCK0

Вывод	Сигнал
V21	VDDI
V22	C3D5
V23	C3D4
V24	VSS
V25	C3D3
V26	C3D2
V3	VSS
V4	VDDE2
V5	XCK1
V6	VDDE2
W1	XCS0
W2	VDDE2
W21	XC3STRB
W22	VSS
W23	C3D1
W24	C3D0
W25	VDDE
W26	XC3HOLDI
W3	XCS1
W4	ODT0
W5	VSS
W6	ODT1
Y1	XRAS
Y2	XCAS
Y21	VDDI
Y22	C2IS
Y23	VDDE
Y24	C3IS
Y25	U1XNMI
Y26	VSS
Y3	VSS
Y4	VDDE2
Y5	CKE0
Y6	CKE1

10.2 Конструктивные характеристики

Микросхема изготавливается в 544-выводном пластиковом корпусе типа High Speed Ball Grid Array (HSBGA-544L). Внешний вид корпуса микросхемы представлен на Рисунок 10.2 с указанием сведений о габаритных и установочных размеров. Особенностью данного корпуса является наличие медной пластины на верхней поверхности корпуса, к которой приклеивается кристалл микросхемы. Это существенно понижает тепловое сопротивление корпуса.

Основные характеристики корпуса:

Механические характеристики:

- Тип корпуса – HSBGA-544L;
- Габаритные размеры корпуса – 27x27x2,36 мм;
- Шаг выводов – 1,00 мм;
- Размер вывода – 0,70 мм;
- Масса не более – 5,00 г.

Тепловые характеристики:

- Тепловое сопротивление при скорости обдува 0 м/с – 14,8 °С/Вт;
- Тепловое сопротивление при скорости обдува 1 м/с – 12,7 °С/Вт;
- Тепловое сопротивление при скорости обдува 2 м/с – 12,0 °С/Вт.

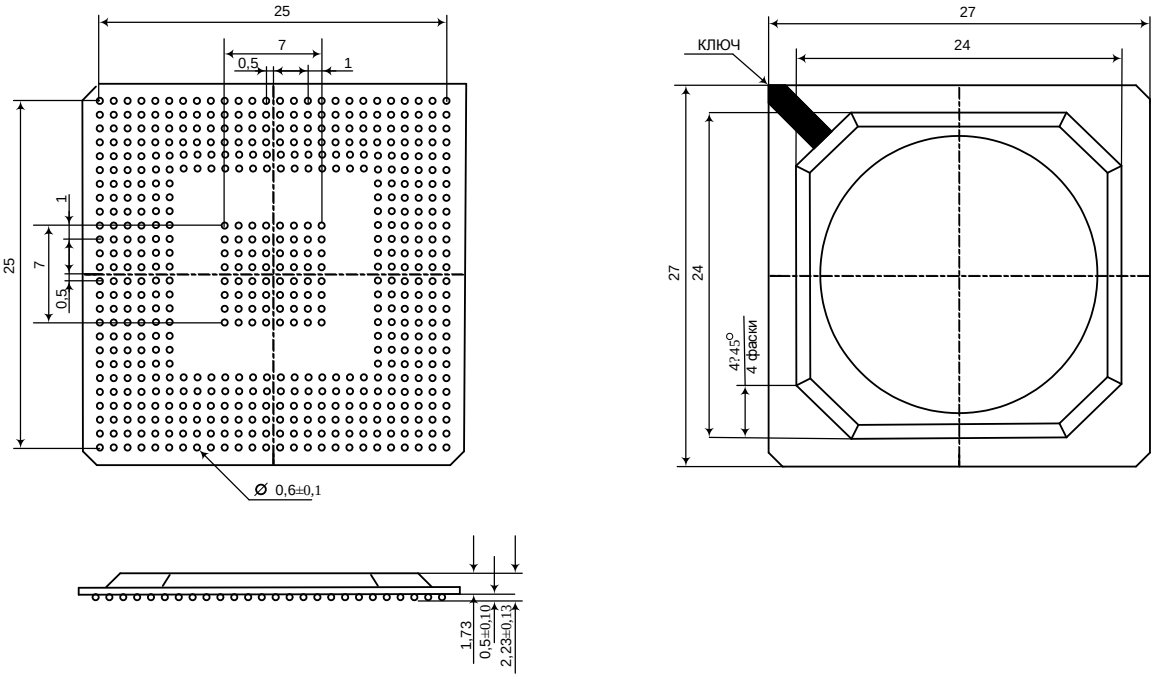


Рисунок 10.2 - Корпус микросхемы

10.3 Электрические характеристики

Таблица 10.4 - Предельные режимы работы

Обозначение	Условие / характеристика	Диапазон значений		Единица измерения
		не менее	не более	
U_{CC1}	Напряжение питания КМОП буферов ввода/вывода	минус 0,5	4,6	В
U_{CC2}	Напряжение питания SSTL буферов ввода/вывода	минус 0,5	2,5	В
U_{CC3}	Напряжение питания ядра	минус 0,5	1,3	В
U_{IH}	Входное напряжение высокого уровня	-	-	В
U_{IL}	Входное напряжение низкого уровня	-	-	В
U_{IN}	Напряжение на КМОП входах	минус 0,5	4,6	В
U_{OUT}	Напряжение, приложенное к выходу	минус 0,5	4,6	В
I_{OUT}	Выходной ток 4-миллиамперный выход 6-миллиамперный выход 8-миллиамперный выход	 минус 16 минус 16 минус 16	 +16 +16 +16	мА
T	Температура окружающей среды	минус 55	125	°С

Примечания 1 Внешние воздействия, значения которых выходят за пределы указанных диапазонов, могут приводить к выходу из строя микросхемы.
2 Значения всех напряжений приведены относительно выводов земли.

Таблица 10.5 – Предельно-допустимые условия эксплуатации

Обозначение	Условие / характеристика	Диапазон значений		Единица измерения
		не менее	не более	
U_{CC1}	Напряжение питания КМОП буферов ввода/вывода	3,0	3,6	В
U_{CC2}	Напряжение питания SSTL буферов ввода/вывода	1,65	1,95	В
U_{CC3}	Напряжение питания ядра	0,95	1,05	В
U_{IH}	Входное напряжение высокого уровня	2,0	3,9	В
U_{IL}	Входное напряжение низкого уровня	минус 0,3	0,8	В
U_{IN}	Напряжение на КМОП входах	минус 0,3	4,0	В
U_{OUT}	Напряжение, приложенное к выходу	минус 0,3	4,0	В
I_{OUT}	Выходной ток 4-миллиамперный выход 6-миллиамперный выход 8-миллиамперный выход	 минус 4 минус 6 минус 8	 +4 +6 +8	мА
VBUS	Питание 5В USB	0	5,25	В
T	Температура окружающей среды	минус 45	85	°С

Примечания 1 Внешние воздействия, значения которых выходят за пределы указанных диапазонов, могут приводить к выходу из строя микросхемы.
2 Значения всех напряжений приведены относительно выводов земли.

Таблица 10.6 - Статические электрические характеристики микросхемы интегральной 1879ВМ6Я

Обозначение	Параметр	Условия измерения	Диапазон значений			Температура среды, °С
			не менее	номинал	не более	
U_{OL1}	Напряжение низкого уровня на выходах КМОП буферов	I_{OL1} = 100 мкА U_{CC1} = 3,6 В	0	-	0,2	-45 - +85
U_{OH1}	Напряжение высокого уровня на выходах КМОП буферов	I_{OH1} = -100 мкА U_{CC1} = 3,0 В	2,8	-	3,0	-45 - +85
U_{OL2}	Напряжение низкого уровня на выходах SSTL буферов	I_{OL2} = 11 мА U_{CC2} = 1,95 В	0	-	0,28	-45 - +85
U_{OH2}	Напряжение высокого уровня на выходах SSTL буферов	I_{OH2} = -11 мА U_{CC2} = 1,65 В	1,67		1,95	-45 - +85
I_{LIL1}	Ток утечки низкого уровня на входе КМОП буфера	U_{IL} = 0 В, U_{CC1} = 3,6В		-	10	-45 - +85
I_{LIH1}	Ток утечки высокого уровня на входе КМОП буфера	U_{IL} = 3,6 В, U_{CC1} = 3,6 В		-	10	-45 - +85
I_{LIL2}	Ток утечки низкого уровня на входе SSTL буфера	U_{CC2} = 1,95 В		-	10	-45 - +85
I_{LIH2}	Ток утечки высокого уровня на входе SSTL буфера	U_{CC2} = 1,95 В		-	10	-45 - +85

Примечания
1 Значения всех напряжений приведены относительно земли.
2 Для входов с “pull-up” резистором ток утечки может превышать указанную величину.

Таблица 10.7 - Динамические электрические характеристики микросхемы интегральной 1879ВМ6Я

Обозначение	Параметр	Условия измерения	Диапазон значений			Единица измерения
			не менее	Номинал	не более	
I_{CC1}	Ток потребления КМОП периферии по входам VDDE	U_{CC1} =(3,3±0,3) В	-	-	500	мА
I_{CC2}	Ток потребления SSTL периферии по входам VDDE2	F_{CLKDDR} =400МГц U_{CC2} =(1,8±0,15) В	-	-	1000	мА
I_{CC3}	Ток потребления процессорного ядра по входам VDDI	F_{CLK}=500 МГц U_{CC3} =(1,0±0,05) В	-	-	1700	мА

10.4 Временные характеристики

В данном подразделе приведены временные диаграммы и временные параметры сигналов микросхемы. Временные параметры микросхемы определялись в полном диапазоне внешних воздействий (см. Таблица 10.8).

10.4.1 Временные диаграммы и временные параметры тактовых сигналов и сигналов общего назначения

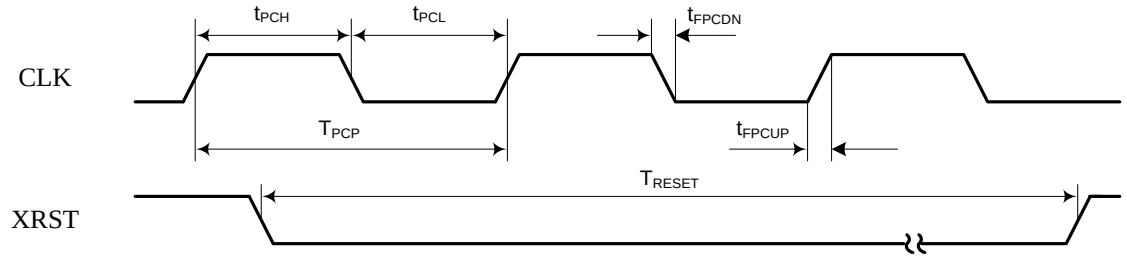


Рисунок 10.3 - Временная диаграмма тактового сигнала и сигнала сброса процессора

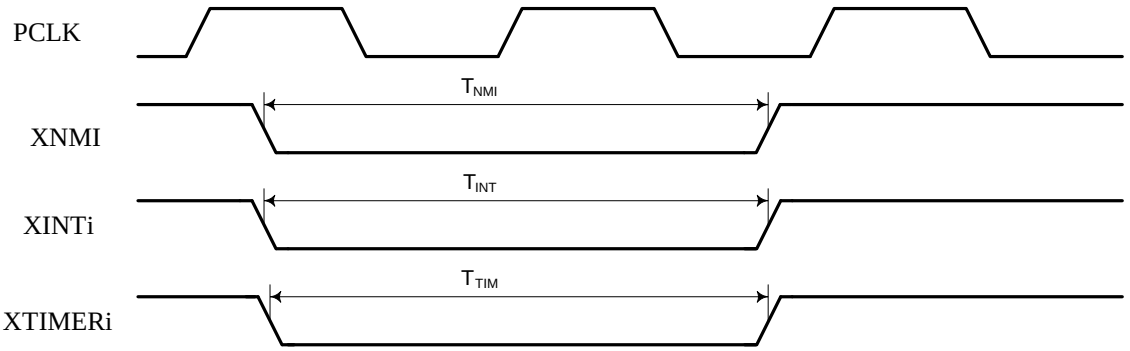


Рисунок 10.4 - Временная диаграмма входов прерываний и входов таймеров процессора

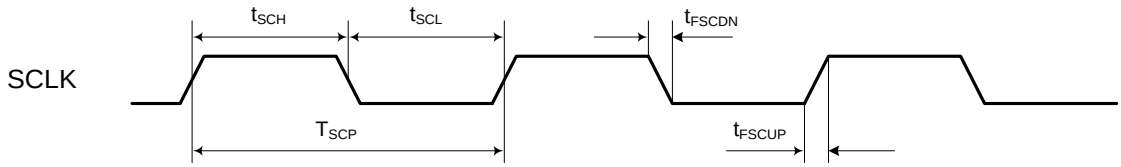


Рисунок 10.5 - Временная диаграмма тактового сигнала внешней шины

Таблица 10.8 - Временные параметры тактовых сигналов и входных сигналов общего назначения

Обозначение	Функциональное описание	Временной параметр, нс	
		не менее	не более
T_{PCP}	Период сигнала тактового сигнала процессора PCLK	80	100
t_{PCH}	Длительность сигнала высокого уровня на входе PCLK	40 % периода тактового сигнала	60 % периода тактового сигнала
t_{PCL}	Длительность сигнала низкого уровня на входе PCLK	40 % периода тактового сигнала	60 % периода тактового сигнала
t_{FPCUP}	Длительность фронта сигнала PCLK		0,2
t_{FPCDN}	Длительность среза сигнала PCLK		0,2
T_{RESET}	Длительность сигнала системного сброса процессора	50∗P	
T_{NMI}	Длительность сигнала низкого уровня на входе немаскируемого прерывания	1,5∗P	
T_{INT}	Длительность сигнала низкого уровня на входе маскируемых прерываний	1,5∗P	
T_{TIMER}	Длительность сигнала низкого уровня на входе таймеров процессора	1,5∗P	
T_{SCP}	Период сигнала тактового сигнала внешних шин процессора SCLK	80	160
t_{SCH}	Длительность сигнала высокого уровня на входе SCLK	40 % периода тактового сигнала	60 % периода тактового сигнала
t_{SCL}	Длительность сигнала низкого уровня на входе SCLK	40 % периода тактового сигнала	60 % периода тактового сигнала
t_{FSCUP}	Длительность фронта сигнала SCLK		0,2
t_{FSCDN}	Длительность среза сигнала SCLK		0,2

Примечание- Параметр P равен периоду выбранного тактового сигнала процессора.

10.4.2 Временные диаграммы и временные параметры сигналов при обмене данными по коммуникационному порту

Временные диаграммы и временные параметры работы коммуникационных портов процессора идентичны, поэтому ниже представлены обобщенные диаграммы работы коммуникационного порта.

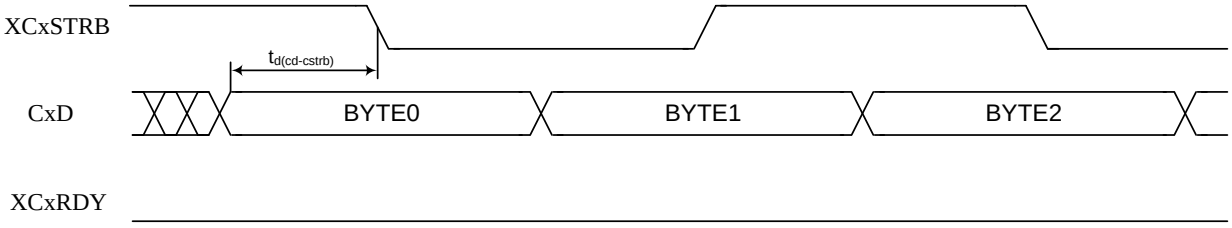


Рисунок 10.6 - Временные диаграммы передачи данных по коммуникационному порту

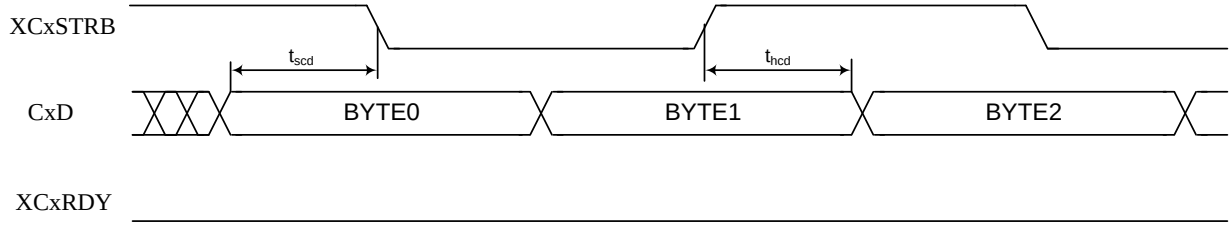


Рисунок 10.7 - Временные диаграммы приема данных по коммуникационному порту

Таблица 10.9 - Временные параметры сигналов при обмене данными по коммуникационному порту

Обозначение	Функциональное описание	Значение, нс	
		не менее	не более
$t_{d(CD\text{-}CSTRB)}$	Задержка выдачи сигнала $\overline{CxSTRB}$ относительно выдачи данных	N*P – 0.8	N*P + 0.8
t_{sCD}	Предустанов принимаемых данных относительно фронта сигнала $\overline{CxSTRB}$	0.5	
t_{hCD}	Удержание принимаемых данных относительно фронта сигнала $\overline{CxSTRB}$	0.4	

Примечания

1 При обозначении выводов символ “x” для коммуникационного порта COM0 принимает значение 0, а для коммуникационного порта COM1 принимает значение 1.

2 Параметр P равен периоду внутреннего тактового сигнала процессора. Период внутреннего тактового сигнала процессора равен $T_{PCLK}/40$.

3 Параметр N зависит от режима работы коммуникационного порта:
N = 2 – скорость 125 МБ/с
N = 2,5 – скорость 100 МБ/с
N = 3 – скорость 83 МБ/с.

Приложение А Программа инициализации контроллера EMI

(справочное)

```
//#########################################################################
const EMIC_BASE         = 1002_0000h;
const PHY_BASE          = 1002_0080h;
const EMI_BASE          = 2000_0000h;

data values

      EMIC_Init_data    : word [66] = (
00000000_00000000_00000100_00000000b, //   DENALI_CTL_00     VERSION:RD:16:16
                                      //   DRAM_CLASS:RW:8:4 START:RW:0:1
00000000_00000000_00000000_00000000b, //   DENALI_CTL_01
                                      //   READ_DATA_FIFO_DEPTH:RD:24:8
                                      //   MAX_CS_REG:RD:16:2
                                      //   MAX_COL_REG:RD:8:4
                                      //   MAX_ROW_REG:RD:0:4
00000000_00000000_00000000_00000000b, //   DENALI_CTL_02
                                      //   ASYNC_CDC_STAGES:RD:24:8
                                      //   WRITE_DATA_FIFO_PTR_WIDTH:RD:16:8
                                      //   WRITE_DATA_FIFO_DEPTH:RD:8:8
                                      //   READ_DATA_FIFO_PTR_WIDTH:RD:0:8
00000000_00000000_00000000_00000000b, //   DENALI_CTL_03
//   AXI0_WRCMD_PROC_FIFO_LOG2_DEPTH:RD:24:8
//   AXI0_WRFIFO_LOG2_DEPTH:RD:16:8
//   AXI0_RDFIFO_LOG2_DEPTH:RD:8:8 AXI0_CMDFIFO_LOG2_DEPTH:RD:0:8
00000010_00000000_00000000_01010000b, //   DENALI_CTL_04     INITAREF:RW:24:4
                                      //   TINIT:RW:0:24
00000000_11001000_00000000_10100000b, //   DENALI_CTL_05     TDLL:RW:16:16
                                      //   TCPD:RW:0:16
00000000_00000101_00001100_00000000b, //   DENALI_CTL_06
                                      //   ADDITIVE_LAT:RW:24:3 WRLAT:RW:16:4
                                      //   CASLAT_LIN:RW:8:4
                                      //   NO_CMD_INIT:RW:0:1
00010110_00000100_00000010_00000010b, //   DENALI_CTL_07     TRC:RW:24:8
                                      //   TRRD:RW:16:8 TCCD:RW:8:5
                                      //   TBST_INT_INTERVAL:RW:0:3
00010010_00000110_00000011_00010000b, //   DENALI_CTL_08     TFAW:RW:24:8
                                      //   TRP:RW:16:8 TWTR:RW:8:6
                                      //   TRAS_MIN:RW:0:8
00000000_00000101_00000010_00000011b, //   DENALI_CTL_09     TMOD:RW:16:8
                                      //   TMRD:RW:8:5 TRTP:RW:0:3
00000011_00000011_01101101_01100000b, //   DENALI_CTL_10     TCKESR:RW:24:8
                                      //   TCKE:RW:16:3 TRAS_MAX:RW:0:16
00000000_00000110_00000110_00000000b, //   DENALI_CTL_11     AP:RW:24:1
                                      //   TWR:RW:16:6 TRCD:RW:8:8
                                      //   WRITEINTERP:RW:0:1
00000010_00001100_00000001_00000001b, //   DENALI_CTL_12     BSTLEN:RW_D:24:3
                                      //   TDAL:RW:16:6 TRAS_LOCKOUT:RW:8:1
                                      //   CONCURRENTAP:RW:0:1
00000000_00000000_00000000_00000111b, //   DENALI_CTL_13     RESERVED:RW:24:1
                                      //   AREFRESH:WR:16:1
                                      //   REG_DIMM_ENABLE:RW:8:1
                                      //   TRP_AB:RW:0:8
00000000_00000000_01001110_00000001b, //   DENALI_CTL_14     TRFC:RW:8:10
                                      //   TREF_ENABLE:RW:0:1
00000000_00000000_00001100_00101101b, //   DENALI_CTL_15
```

```
                                          //   POWER_DOWN:RW:24:1 TREF:RW:0:14
    00000000_11001000_00000000_00000010b, //   DENALI_CTL_16     TXSR:RW:16:16
                                          //   TPDEX:RW:0:16
    00000000_00000000_00000000_00110111b, //   DENALI_CTL_17
                                          //   PWRUP_SREFRESH_EXIT:RW:24:1
                                          //   SREFRESH:RW+:16:1 TXSNR:RW:0:16
    00000000_00000000_00000001_00000000b, //   DENALI_CTL_18     RESERVED:RW:24:5
                                          //   CKE_DELAY:RW:16:3
                                          //   ENABLE_QUICK_SREFRESH:RW:8:1
                                          //   SREFRESH_EXIT_NO_REFRESH:RW:0:1
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_19
                                          //   WRITE_MODEREG:RW+:0:26
    00000000_00001010_01100010_00000000b, //   DENALI_CTL_20
                                          //   MR0_DATA_0:RW:8:15
                                          //   MRW_STATUS:RD:0:8
    00000000_00000000_00000000_00000100b, //   DENALI_CTL_21
                                          //   MR2_DATA_0:RW:16:15
                                          //   MR1_DATA_0:RW:0:15
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_22
                                          //   MR3_DATA_0:RW:16:15
                                          //   MRSINGLE_DATA_0:RW:0:15
    00000000_00000100_00001010_01100010b, //   DENALI_CTL_23
                                          //   MR1_DATA_1:RW:16:15
                                          //   MR0_DATA_1:RW:0:15
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_24
                                          //   MRSINGLE_DATA_1:RW:16:15
                                          //   MR2_DATA_1:RW:0:15
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_25
                                          //   BIST_RESULT:RD:24:2
                                          //   BIST_GO:WR:16:1 MR3_DATA_1:RW:0:15
    00000000_00000001_00000001_00000000b, //   DENALI_CTL_26
                                          //   BIST_ADDR_CHECK:RW:16:1
                                          //   BIST_DATA_CHECK:RW:8:1
                                          //   ADDR_SPACE:RW:0:6
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_27
                                          //   BIST_START_ADDRESS:RW:0:32
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_28
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_29
                                          //   BIST_DATA_MASK:RW:0:32
    00001010_00000000_00000001_00000000b, //   DENALI_CTL_30
                                          //   APREBIT:RW_D:24:4 COL_DIFF:RW:16:4
                                          //   ROW_DIFF:RW:8:3 BANK_DIFF:RW:0:2
    00000001_00000001_11111111_11111111b, //   DENALI_CTL_31     RESERVED:RW:24:1
                                          //   ADDR_CMP_EN:RW:16:1
                                          //   COMMAND_AGE_COUNT:RW:8:8
                                          //   AGE_COUNT:RW:0:8
    00000001_00000001_00000001_00000001b, //   DENALI_CTL_32
                                          //   RW_SAME_EN:RW:24:1
                                          //   PRIORITY_EN:RW:16:1
                                          //   PLACEMENT_EN:RW:8:1
                                          //   BANK_SPLIT_EN:RW:0:1
    00000001_00000001_00000001_00000001b, //   DENALI_CTL_33
    //   DISABLE_RW_GROUP_W_BNK_CONFLICT:RW:24:2 W2R_SPLIT_EN:RW:16:1
    //   CS_SAME_EN:RW:8:1 RW_SAME_PAGE_EN:RW:0:1
    00000000_00000000_00000001_00001011b, //   DENALI_CTL_34
                                          //   INHIBIT_DRAM_CMD:RW:24:1
                                          //   DISABLE_RD_INTERLEAVE:RW:16:1
                                          //   SWAP_EN:RW:8:1
                                          //   NUM_Q_ENTRIES_ACT_DISABLE:RW:0:4b
    00000000_00000000_00000000_00000011b, //   DENALI_CTL_35
```

```
                                        //   IN_ORDER_ACCEPT:RW:24:1
                                        //   Q_FULLNESS:RW:16:4 REDUC:RW:8:1
                                        //   CS_MAP:RW:0:2
    00000000_00000001_00000000_00000000b, //   DENALI_CTL_36
                                        //   CTRLUPD_REQ_PER_AREF_EN:RW:16:1
                                        //   CTRLUPD_REQ:WR:8:1
                                        //   CONTROLLER_BUSY:RD:0:1
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_37     INT_ACK:WR:16:11
                                        //   INT_STATUS:RD:0:12
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_38     INT_MASK:RW:0:12
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_39
                                        //   OUT_OF_RANGE_ADDR:RD:0:32
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_40
                                        //   OUT_OF_RANGE_TYPE:RD:16:6
                                        //   OUT_OF_RANGE_LENGTH:RD:8:7
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_41
                                        //   OUT_OF_RANGE_SOURCE_ID:RD:0:10
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_42
                                        //   BIST_EXP_DATA:RD:0:64
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_43
                                        //   BIST_EXP_DATA:RD:0:64
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_44
                                        //   BIST_FAIL_DATA:RD:0:64
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_45
                                        //   BIST_FAIL_DATA:RD:0:64
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_46
                                        //   BIST_FAIL_ADDR:RD:0:32
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_47
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_48
                                        //   PORT_CMD_ERROR_ADDR:RD:0:32
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_49
                                        //   PORT_CMD_ERROR_TYPE:RD:24:2
                                        //   PORT_CMD_ERROR_ID:RD:8:10
    00000010_00000010_00000001_00000001b, //   DENALI_CTL_50
                                        //   ODT_WR_MAP_CS1:RW:24:2
                                        //   ODT_RD_MAP_CS1:RW:16:2
                                        //   ODT_WR_MAP_CS0:RW:8:2
                                        //   ODT_RD_MAP_CS0:RW:0:2
    00000001_00000011_00000011_00000011b, //   DENALI_CTL_51     ODT_EN:RW:24:1
                                        //   TODTH_RD:RW:16:4 TODTH_WR:RW:8:4
                                        //   TODTL_2CMD:RW:0:4
    00000000_00000000_00000011_00000010b, //   DENALI_CTL_52
                                        //   RD_TO_ODTH:RW:8:5
                                        //   WR_TO_ODTH:RW:0:5
    00000010_00000010_00000000_00000000b, //   DENALI_CTL_53
                                        //   R2W_DIFFCS_DLY:RW_D:24:5
                                        //   R2R_DIFFCS_DLY:RW_D:16:5
    00000010_00000000_00000010_00000010b, //   DENALI_CTL_54
                                        //   R2W_SAMECS_DLY:RW_D:24:5
                                        //   R2R_SAMECS_DLY:RW:16:5
                                        //   W2W_DIFFCS_DLY:RW_D:8:5
                                        //   W2R_DIFFCS_DLY:RW_D:0:5
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_55
                                        //   OCD_ADJUST_PUP_CS_0:RW:24:5
                                        //   OCD_ADJUST_PDN_CS_0:RW:16:5
                                        //   W2W_SAMECS_DLY:RW:8:5
                                        //   W2R_SAMECS_DLY:RW:0:5
    00000000_00000000_00000100_00000100b, //   DENALI_CTL_56
                                        //   CKE_STATUS:RD:24:2
                                        //   AXI0_FIFO_TYPE_REG:RW:16:2
```

```
                                        //   AXI0_W_PRIORITY:RW:8:3
                                        //   AXI0_R_PRIORITY:RW:0:3
    00000000_00000000_00000000_00000000b, //   DENALI_CTL_57
                                        //   TDFI_PHY_WRLAT:RD:24:4
                                        //   DLL_RST_ADJ_DLY:RW:16:8
                                        //   DLL_RST_DELAY:RW:0:16
    00000000_00000000_00001000_00000000b, //   DENALI_CTL_58
                                        //   DRAM_CLK_DISABLE:RW:24:2
                                        //   TDFI_RDDATA_EN:RD:16:4
                                        //   TDFI_PHY_RDLAT:RW_D:8:4
                                        //   UPDATE_ERROR_STATUS:RD:0:7
    00000000_00011000_01011010_00000000b, //   DENALI_CTL_59
                                        //   TDFI_CTRLUPD_MAX:RW:8:14
                                        //   TDFI_CTRLUPD_MIN:RD:0:4
    00000010_00000000_00000010_00000000b, //   DENALI_CTL_60
                                        //   TDFI_PHYUPD_TYPE1:RW:16:16
                                        //   TDFI_PHYUPD_TYPE0:RW:0:16
    00000010_00000000_00000010_00000000b, //   DENALI_CTL_61
                                        //   TDFI_PHYUPD_TYPE3:RW:16:16
                                        //   TDFI_PHYUPD_TYPE2:RW:0:16
    00000000_00000000_00011000_01011010b, //   DENALI_CTL_62
                                        //   TDFI_PHYUPD_RESP:RW:0:14
    00000000_00000000_01111001_11000010b, //   DENALI_CTL_63
                                        //   TDFI_CTRLUPD_INTERVAL:RW:0:32
    00000000_00000001_00000101_00000101b, //   DENALI_CTL_64     RESERVED:RW:24:5
                                        //   ODT_ALT_EN:RW:16:1
                                        //   WRLAT_ADJ:RW:8:4 RDLAT_ADJ:RW:0:4
    00000000_00000000_00000000_00000000b  //   DENALI_CTL_65     RESERVED:RW:0:1
                                             );
          PHY_Init_data     : word [72] = (
    00000000_00000000_00000100_00010010b, // DEN_PHY_DQ_TIMING_REG_0:RW:0:32
    00000000_00000000_00000100_00010100b, // DEN_PHY_DQS_TIMING_REG_0:RW:0:32
    00100000_00000000_00000000_01101000b, // DEN_PHY_GATE_LPBK_CTRL_REG_0:RW:0:32
    00000000_00000000_00000000_00000100b, // DEN_PHY_READ_CTRL_REG_0:RW:0:32
    00000000_00010010_00000000_00100100b, // PHY_DLL_MASTER_CTRL_REG_0:RW:0:32
    01000000_01000000_01000000_01000000b, // PHY_DLL_SLAVE_CTRL_REG_0:RW:0:32
    00000000_00000000_01001001_00100001b, // DEN_PHY_IE_TIMING_REG_0:RW:0:32
    00000000_00000000_00000000_00000000b, // DEN_PHY_OBS_REG_0_0:RD:0:32
    00000000_00000000_00000000_00000000b, // PHY_DLL_OBS_REG_0_0:RD:0:32
    00000000_00000000_00000000_00000000b, // PHY_DLL_OBS_REG_1_0:RD:0:32
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000100_00010010b, // DEN_PHY_DQ_TIMING_REG_0:RW:0:32
    00000000_00000000_00000100_00010100b, // DEN_PHY_DQS_TIMING_REG_0:RW:0:32
    00100000_00000000_00000000_01101000b, // DEN_PHY_GATE_LPBK_CTRL_REG_0:RW:0:32
    00000000_00000000_00000000_00000100b, // DEN_PHY_READ_CTRL_REG_0:RW:0:32
    00000000_00010010_00000000_00100100b, // PHY_DLL_MASTER_CTRL_REG_0:RW:0:32
    01000000_01000000_01000000_01000000b, // PHY_DLL_SLAVE_CTRL_REG_0:RW:0:32
    00000000_00000000_01001001_00100001b, // DEN_PHY_IE_TIMING_REG_0:RW:0:32
    00000000_00000000_00000000_00000000b, // DEN_PHY_OBS_REG_0_0:RD:0:32
    00000000_00000000_00000000_00000000b, // PHY_DLL_OBS_REG_0_0:RD:0:32
    00000000_00000000_00000000_00000000b, // PHY_DLL_OBS_REG_1_0:RD:0:32
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
```

```
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000100_00010010b, // DEN_PHY_DQ_TIMING_REG_0:RW:0:32
    00000000_00000000_00000100_00010100b, // DEN_PHY_DQS_TIMING_REG_0:RW:0:32
    00100000_00000000_00000000_01101000b, // DEN_PHY_GATE_LPBK_CTRL_REG_0:RW:0:32
    00000000_00000000_00000000_00000100b, // DEN_PHY_READ_CTRL_REG_0:RW:0:32
    00000000_00010010_00000000_00100100b, // PHY_DLL_MASTER_CTRL_REG_0:RW:0:32
    01000000_01000000_01000000_01000000b, // PHY_DLL_SLAVE_CTRL_REG_0:RW:0:32
    00000000_00000000_01001001_00100001b, // DEN_PHY_IE_TIMING_REG_0:RW:0:32
    00000000_00000000_00000000_00000000b, // DEN_PHY_OBS_REG_0_0:RD:0:32
    00000000_00000000_00000000_00000000b, // PHY_DLL_OBS_REG_0_0:RD:0:32
    00000000_00000000_00000000_00000000b, // PHY_DLL_OBS_REG_1_0:RD:0:32
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000100_00010010b, // DEN_PHY_DQ_TIMING_REG_0:RW:0:32
    00000000_00000000_00000100_00010100b, // DEN_PHY_DQS_TIMING_REG_0:RW:0:32
    00100000_00000000_00000000_01101000b, // DEN_PHY_GATE_LPBK_CTRL_REG_0:RW:0:32
    00000000_00000000_00000000_00000100b, // DEN_PHY_READ_CTRL_REG_0:RW:0:32
    00000000_00010010_00000000_00100100b, // PHY_DLL_MASTER_CTRL_REG_0:RW:0:32
    01000000_01000000_01000000_01000000b, // PHY_DLL_SLAVE_CTRL_REG_0:RW:0:32
    00000000_00000000_01001001_00100001b, // DEN_PHY_IE_TIMING_REG_0:RW:0:32
    00000000_00000000_00000000_00000000b, // DEN_PHY_OBS_REG_0_0:RD:0:32
    00000000_00000000_00000000_00000000b, // PHY_DLL_OBS_REG_0_0:RD:0:32
    00000000_00000000_00000000_00000000b, // PHY_DLL_OBS_REG_1_0:RD:0:32
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000000_00000000_00000000_00000000b, //
    00000001_00000000_00000101b, // DEN_PHY_CTRL_REG:RW:0:32
    00000000_00000000_00000000_00000000b, // DEN_PHY_PAD_TSEL_REG:RW:0:32
    11111111_11111011_11111111_11111011b, // PHY_PAD_DRIVE_REG_0:RW:0:32
    11111111_11111011_11111111_11111011b, // PHY_PAD_DRIVE_REG_1:RW_D+:0:32
    11111111_11111011_11111111_11111011b, // PHY_PAD_DRIVE_REG_2:RW:0:32
    00000000_00000000_00000000_00000000b, // PHY_PAD_TERM_REG_0:RW_D+:0:32
    00000000_00000000_00000000_00000000b, // PHY_PAD_TERM_REG_1:RW_D+:0:32
    00000000_00000000_00000000_00000000b  // PHY_PAD_TERM_REG_2:RW:0:32
                                          );

    end values;
```

```
//################################## INTERRUPT VECTORS#########################
begin text_START_INT
      goto mytest;
      nul;
      nul;
      nul;
      nul;
end text_START_INT;

begin text_EMIC_INT
      goto EMIC_Handler;
      nul;
      nul;
      nul;
      nul;
end text_EMIC_INT;

//#######################MAIN PROGRAM#############################################
begin text_PROGRAM
      weak mytest       : label;

      local main        : label;
      local Normal      : label;
      local Finish      : label;

      <mytest>
            gr0 = 0h;   //    arg 1
            gr1 = 0h;   //    arg 2
            gr2 = 0h;   //    arg 3
            gr3 = 0h;   //    counter 1
            gr4 = 0h;   //    counter 2
            gr5 = 0h;   //    common
            gr6 = 0h;   //    common
            gr7 = 0h;   //    common

            ar0 = 0h;
            ar1 = 0h;
            ar2 = 0h;
            ar3 = 0h;
            ar4 = 0h;
            ar5 = 0h;
            ar6 = 0h;
            ar7 = 0001_0000h; //    stack pointer

      <main>
            call EMIC_Initialisation;
            call EMI_RW_test;
            goto Normal;
      goto main;

//############################## EMIC initialization ##########################
      <EMIC_Initialisation>
//############################## Enable NMU interrupts ########################
            pswr set 0040h;
            gr0 = 0040_0000h;
            ar0 = 4000_0408h;
            [ar0] = gr0;
            gr2 = intr;
```

```
//######################## Config. registers write ####################
            gr3 = 66;
            ar0 = EMIC_BASE;
            ar1 = EMIC_Init_data [0];

      <EMIC_Init_cycle>
            gr0 = [ar1++];
            [ar0++] = gr0;
            gr3 = gr3 - 1;
            if <>0 goto EMIC_Init_cycle;
//######################## PHY registers write ########################
            ar0 = PHY_BASE;
            gr3 = 72;
            ar1 = PHY_Init_data [0];
      <PHY_Init_cycle>
            gr0 = [ar1++];
            [ar0++] = gr0;
            gr3 = gr3 - 1;
            if <>0 goto PHY_Init_cycle;
//######################## EMIC Start ###################################
            ar0 = EMIC_BASE;
            gr0 = 0000_0401h;
            [ar0] = gr0;
//######################## Waiting end of init ########################
            gr1 = 0000_0008h;
      <EMIC_Init_wait>
            gr0 = [ar0 + 37];
            gr0 = gr0 and gr1;
            if =0 goto EMIC_Init_wait;
//######################## Clear Interruption bit #####################
            ar0 = EMIC_BASE;
            gr0 = 0008_0000h;
            [ar0 + 37] = gr0;
      return;
//#############################################################################

//#################### Read/Write test ################################
      <EMI_RW_test>
            ar0 = EMI_BASE;
            gr1 = 0;
            gr3 = 1024;
      <Write_read_cycle_0>
            [ar0] = ar0;
            gr0 = [ar0];
            gr2 = ar0;
            gr0 - gr2;
            if <>0 goto write_read_failed;
            gr1 = gr1 + 1;                //    increase GR1 if test passed
      <write_read_failed>
            ar0++;
            gr3 = gr3 - 1;
            if <>0 goto Write_read_cycle_0;
      return;
//#############################################################################
```

```
//############################## Finish algorithm ########################
      <Normal>
            ar0 = 1003_3fffh;
            gr1 = 0_ffff_ffffh;
            [ar0] = gr1;
            goto Finish;
      <Finish>
            halt;

//############################## Count Subroutine ########################
      <Delay_cycle>
            gr1 = gr1 - 1;
            if <>0 goto Delay_cycle;
      return;

//############################## EMIC interrupt handler ##################
      <EMIC_Handler>
            push gr0, ar0;
            push gr1, ar1;
            push gr2, ar2;
            push gr3, ar3;
            push gr4, ar4;
            push gr5, ar5;
            push gr6, ar6;
            push gr7, ar7;

//################## Insert your interrupt program here #####

            pop gr7, ar7;
            pop gr6, ar6;
            pop gr5, ar5;
            pop gr4, ar4;
            pop gr3, ar3;
            pop gr2, ar2;
            pop gr1, ar1;
            pop gr0, ar0;
            ireturn;

//############################################################################

end text_PROGRAM;
```

Лист регистрации изменений

Изм.	Номера страниц				Всего страниц в докумете	№ документа	Входящий № сопроводительного докум. и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					
4		все					ЮФКВ.077-2015	Харламова	13.02.2020