*Список изменений*

|  |  |
| --- | --- |
| **29.12.15** | **Добавлена таблица интерфейса взаимодействия ядра с кэш-памятью** |
| **04.01.16** | **Изменена разрядность сигнала core\_req\_size с 4 до 3 бит** |
| **05.01.16** | **Добавлено описание характеристик кэш-памяти 1 уровня** |
| **07.01.16** | **Добавлено более подробное описание интерфейсов ядра** |
| **11.01.16** | **Добавлен шаблон оглавления** |
| **16.01.16** | **Из интерфейса ядра удалён сигнал core\_req\_be, он будет формироваться в L1 исходя из core\_req\_size и core\_req\_addr** |
| **17.01.16** | **Добавлены временные диаграммы интерфейса ядра с кэшем инструкций** |
| **21.01.16** | **Добавлено описание взаимодействия ядра с кэшем данных** |
| **24.01.16** | **Добавлена блок-схема конвейера ядра** |

Оглавление

[Общее описание системы 1](#_Toc441397996)

[Процессорный кластер 2](#_Toc441397997)

[Описание ядра 2](#_Toc441397998)

[Назначение и место в системе 2](#_Toc441397999)

[Описание конвейера 2](#_Toc441398000)

[Микроархитектура 3](#_Toc441398001)

[Взаимодействие с кэш-памятью данных 4](#_Toc441398002)

[Кэш-память первого уровня 5](#_Toc441398003)

[Периферийный кластер 7](#_Toc441398004)

[Boot Uart 7](#_Toc441398005)

[Программная модель 7](#_Toc441398006)

[SPI 7](#_Toc441398007)

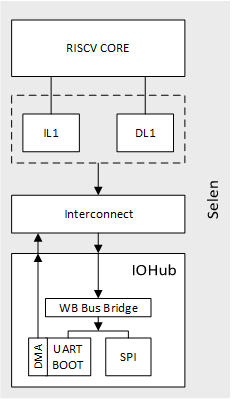
[Программная модель 7](#_Toc441398008)

[Прототип 7](#_Toc441398009)

# ****Общее описание системы****

**Selen – одноядерная микропроцессорная система на кристалле. В состав СнК входят:**

* **32-ух разрядное процессорное ядро с архитектурой системы команд RISC-V;**
* **Кэш-память первого уровня (инструкций и данных);**
* **Контроллер периферийных интерфейсов;**
* **UART, с модулем DMA для загрузки программы с хост-машины во внутреннюю память системы;**
* **SPI;**
* **Коммутатор с интерфейсом Wishbone.**



# ****Процессорный кластер****

**Процессорный кластер состоит из 32-ух битного ядра, поддерживающего систему команд RISCV (см. приложение 1), кэш-памяти инструкций и кэш-памяти данных и блока доступа в память (Memory Access Unit), поддерживающего интерфейсов Wishbone.**

## ****Описание ядра****

## Назначение и место в системе

## Описание конвейера

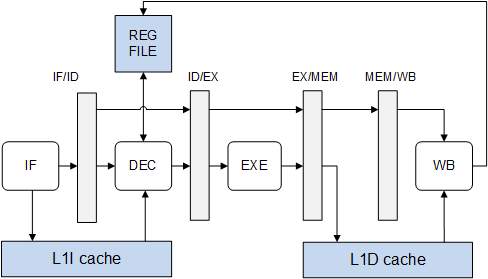


Рисунок 1. Блок схема конвейера

Данный конвейер представляет собой подобие 5 стадийного классического MIPS конвейера. Включаемые стадии описаны ниже. Все операции по вычислению адреса выполняются на стадии execution

## 1) Стадия Instruction fetch (IF)

Выставления запроса в il1 cash для получения инструкции согласно протокол, описанному ниже. Возможные остановки – промах в кэше или полная остановка конвейера (stall).

## 2) Стадия decode (DEC)

Декодирование инструкции выставление соответствующих сигналов управления. Возможные останови – полная остановка конвейера по причинам ожидания Load инструкций или по причине декодирования инструкций типа jump, branch

## 3) Стадия executions (EXE)

Выполнение арифметических и логических операций с операндами(значения из регистра или операнд непосредственно из команды) и операции с адресом(вычислении адреса перехода). Возможные причины остановки – неготовность операндов(ожидание load), полная остановка конвейера.

## 4) Стадия Memory (MEM)

Выполнение обращения в кэшь данных согласно протоколу. Причины остановки – полная остановка конвейера и промах в кэшь данных в случаи load операции.

## 5) Стадия write back (WB)

Выполнение записи данных в регистровый файл. Причины остановки – полная остановка конвейера.

## Микроархитектура

Взаимодействие с подсистемой памяти реализовано согласно протоколу, представленному на рисунке 1.

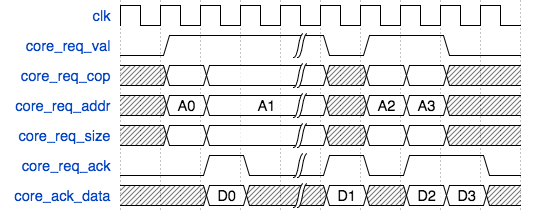


Рисунок 1

Более подробно на рисунке 2. По сбросу регистр счётчика команд сбрасывается на начальное значение (на рисунке это 0). Так же вспомогательный регистр req\_val\_r, сбрасывается в ноль.

После сброса на шину выставляется запрос с адресом равным значения регистра счётчика команд.

Счётчик команд изменяет своё значение когда вспомогательный регистр в 0 или при получения сигнала подтверждения от подсистемы памяти (core\_req\_ack):

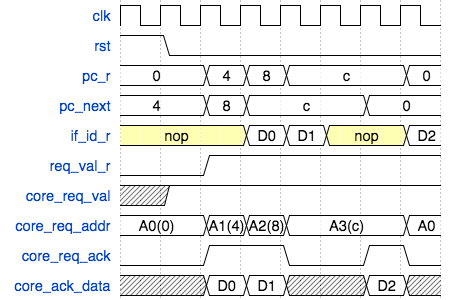
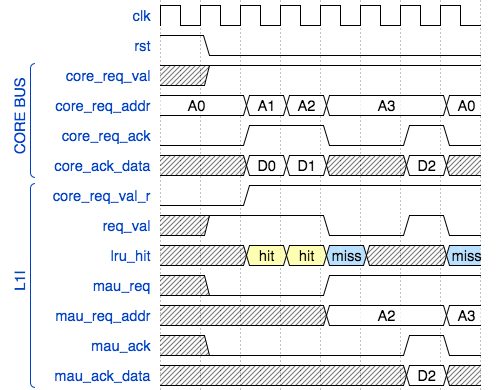


Рисунок 2



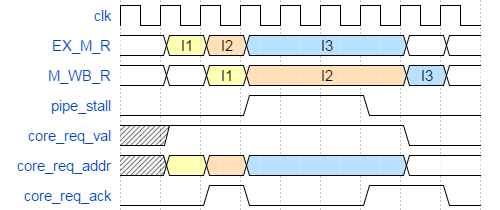
**Таблица 1. Интерфейс взаимодействия ядра с кэш-памятью инструкций**

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Разрядность** | **Направление** | **Описание** |
| **i\_req\_val** | **[0:0]** | **Output** | **Значимость запроса** |
| **i\_req\_addr** | **[31:0]** | **Output** | **Адрес запроса** |
| **i\_ack\_val** | **[0:0]** | **Input** | **Значимость ответа** |
| **i\_ack\_rdata** | **[31:0]** | **Input** | **Данные ответа** |

## ****Взаимодействие с кэш-памятью данных****

На стадии М формируется запрос в память. Адрес запроса берётся с регистра конвейера EX\_M\_R. Инструкция переходит на следующую стадию(WB). На стадии WB из памяти должен вернуться ответ, если он есть то используются полученные данные и конвейер работает в штатном режиме. Если нет, то формируется полная остановка конвейера.

Следует заметить, что когда первая инструкция (I1) переходит со стадии M на стадию WB, на стадию M приходит следующая инструкция (I2). Если это инструкция работы с памятью, то так же выставляется запрос в память. При формирования остановки конвейера, запрос в память от инструкции продолжает висеть.



**Таблица 2. Интерфейс взаимодействия ядра с кэш-памятью данных**

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Разрядность** | **Направление** | **Описание** |
| **d\_req\_val** | **[0:0]** | **Output** | **Значимость запроса** |
| **d\_req\_addr** | **[31:0]** | **Output** | **Адрес запроса** |
| **d\_req\_cop** | **[2:0]** | **Output** | **Типа запроса:**  **3’b000 – RD**  **3’b001 – WR**  **3’b010 – RDNC**  **3’b011 - WRNC**  **Остальные – зарезервировано** |
| **d\_req\_wdata** | **[31:0]** | **Output** | **Данные для записи** |
| **d\_req\_size** | **[2:0]** | **Output** | **Размер запроса (1, 2 или 4 байта)** |
| **d\_ack\_val** | **[0:0]** | **Input** | **Значимость ответа** |
| **d\_ack\_rdata** | **[31:0]** | **Input** | **Данные ответа** |

## Кэш-память первого уровня

Характеристики кэш-памяти инструкций:

* Ассоциативность – 4 канала;
* Объём – 32 KB;
* Длина строки (размер блока данных) – 32 B;
* Общее количество строк – 32 KB / 32B = 1K;
* Алгоритм вытеснения: LRUm;
* Write-through, No-Write-Allocate.

Общая формула расчёт размера выглядит следующим образом: , где, *N* – объём кэш-памяти, *l* – длина кэш-строки, *m* – ассоциативность, *s* – количество сетов, *b* – количество банков.

При *N* = 32 KB, *m* = 4, *l* = 32 B и *b* = 1, s получается равным 256.

При такой структуре памяти, физический адрес состоит из 3 элементов:

* tag – тэг адреса, хранится вместе с кэш-строкой;
* idx – номер сета;
* offset – смещение внутри кэш-строки.

Для предлагаемых параметров и длине физического адреса 32 бита получаем:

* tag = 19 бит;
* idx = 8 бит;
* offset = 5 бита.

Таблица 1. Разбиение физического адреса в L1

|  |  |  |
| --- | --- | --- |
| Тэг | Индекс | Номер байта |
| [31:13] | [12:5] | [4:0] |

Таблица 2. Внутренние проверки

|  |  |
| --- | --- |
| Свойство | Описание |
| offset\_allign\_p |  |
| tag\_cmp\_with\_val\_vect\_p | one hot |
| lru\_way\_vect\_p | one hot |

# ****Периферийный кластер****

## ****Boot Uart****

## ****Программная модель****

## ****SPI****

## Программная модель

# Прототип