**Draft**

16

SELEN

Reference manual

Процессорная система на кристалле Selen: руководство и описание

**https://github.com/Zhikharev/selen**

*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** |
| **29.01.16** | **Добавлено начальное описание конвейера. Добавлены диаграммы взаимодействия с подсистемой памяти** |
| **30.01.2016** | **Добавлено подробное описание 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)

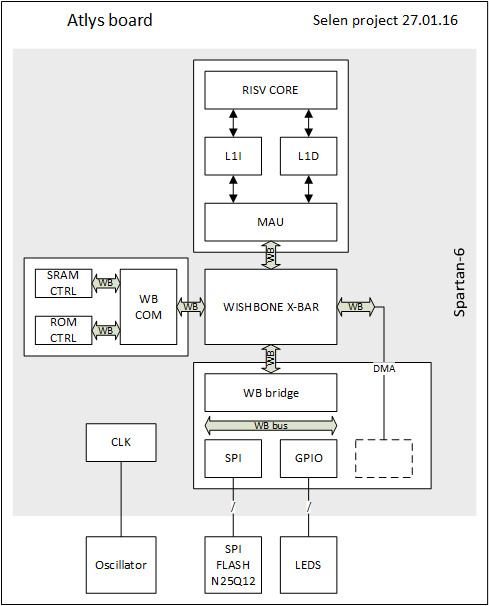
[Программная модель 7](#_Toc441398008)

[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);**



**Рисунок 1**

# ****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

Главное назначение ядра – выполнение программы, загруженной во флэш память системы.

## Описание конвейера

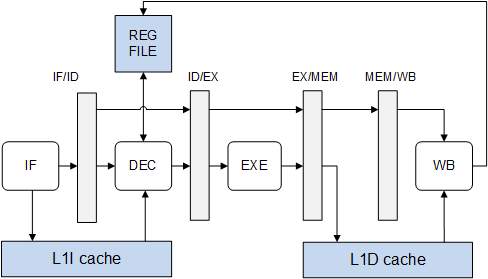


Рисунок 1. Блок схема конвейера

Конвейер состоит из пяти стадий:

1. IF (Instruction Fetch) – стадия подкачки инструкций. На данной стадии происходит выставление запроса в кэш-инструкций и формирование нового значения программного счётчика (NPC). На следующею стадию передается инструкция, текущее значение программного счетчика(current pc) и значение программного счетчика +4(pc next = current pc + 4) Возможные остановки: промах в кэше инструкций или полная остановка конвейера;
2. DEC (Decode) – стадия декодирования. На данной стадии происходит декодирование инструкции и формирование сигналов управления. Кроме того осуществляется доступ к регистрового файлу. На данной стадии может сформироваться сигнал остановки стадий IF и DEC при промахе в кэш-память инструкций (с формирование nop для следующих стадий). На следующею стадию передаются данные описанные в таблице выходных данных со стадии decode
3. EX (Execution) – стадия вычислений. На данной стадии происходит выполнение арифметико-логических операций и операции формирования адресов (адрес доступа в память, адрес перехода). Возможные остановки: полная остановка конвейера. На следующую стадию передаются данные описанные в таблице выходных данных со стадии выполнения .
4. MEM (Memory) – стадия обращения в память. На данной стадии происходит обращение в кэш данных для инструкций доступа в память. Возможные остановки: полная остановка конвейера. Данные передаваемые на следующую стадию описаны в таблице выходные данные со стадии memory.
5. B (Write Back) - стадия записи данных в регистровый файл. На данной стадии происходит формирование записи в регистровый файл. Кроме того может сформироваться сигнал полной остановки конвейере при отсутствии данных из кэш-памяти данных.

Таблица выходных данных со стадии декодирования(decode) инструкций

|  |  |  |
| --- | --- | --- |
| Имя | Разрядность | Описание |
| Шина L1d\_bus | [4:0] | Шина обращения в кэш |
| l1d\_ld\_or\_st\_dec | [0:0] | Распределения типа обращения в память(load or store) |
| l1d\_val\_dec | [0:0] | Валидность данных для кэша данных |
| l1d\_size\_dec | [2:0] | Размер запроса в dl1 кэш |
| Mux\_bus | [4:0] | Шина содержит все управляющие сигналы для мультиплексоров |
| is\_imm\_dec | [0:0] | Управляющий сигнал для мультиплексора, выбирающего между imm или src1(src\_a) |
| is\_pc\_dec | [0:0] | Управляющий сигнал для мультиплексора, выбирающего между pc или scr2(srcb) |
| is\_src1\_dec | [0:0] | Управляющий сигнал для мультиплексора, выбирающего между pc + 4 или src1(srca) |
| is\_otw\_pc\_dec | [0:0] | Управляющий сигнал для мультиплексора, выбирающего между pc и выбором мультиплексора is\_src1(Mux 4 на схеме) |
| is\_mem\_dec | [0:0] | Управляющий сигнал для мультиплексора, выбирающего между результатом с памяти и с ALU |
| sx\_imm\_dec | [31:0] | Операнд принятый из кода команды со знаковым расширением |
| Srca\_dec | [31:0] | Операнд из регистрового файла |
| Srcb\_dec | [31:0] | Операнд из регистрового файла |
| Pc\_dec | [31:0] | Текущее значение программного счетчика |
| pc \_4\_dec | [31:0] | Текущее значение программного счетчика + 4 |
| brnch\_cnd\_dec | [1:0] | Условия ветвления |
| we\_reg\_file\_dec | [0:0] | Разрешение записи в регистровый файл |
| alu\_op\_dec | [3:0] | Код операции для ALU |
| Hazard\_bus\_dec | [16:0] | Шина передачи данных hazard\_ctrl |
| rs1\_dec | [4:0] | Адрес регистра источника 1 |
| rs2\_dec | [4:0] | Адрес регистра источника 2 |
| Rd\_dec | [4:0] | Адрес регистра назначения |

Таблица выходных данных со стадии выполнение (execution)

|  |  |  |
| --- | --- | --- |
| Имя | Разрядность | Описание |
| brnch\_taken\_ex | [0:0] | Результат вычислении условия ветвления |
| alu\_result\_ex | [31:0] | Результат вычисления ALU |
| sx\_imm\_ex | [31:0] | Операнд с кода команды |
| w\_data\_ex(src2) | [31:0] | Данные для записи в память |
| addr\_ex | [31:0] | Адрес перехода |
| pc \_4 | [31:0] | Текущее значение программного счетчика + 4 |
| we\_reg\_file\_dec | [0:0] | Разрешение записи в регистровый файл |
| hz\_bus | [16:0] | Включает в себя значения регистров источников(source) и регистра назначения(destination) для регистрового файла |
| rs1\_ex | [4:0] | Адрес регистра источника 1 |
| rs2\_ex | [4:0] | Адрес регистра источника 2 |
| Rd\_ex | [4:0] | Адрес регистра назначения |
| mux\_bus[5] | [0:0] | Значение сигнала управления для 6 мультиплексора(распределяет поток с памяти и с alu для записи в reg\_file) |

Таблица выходных данных со стадии memory

|  |  |  |
| --- | --- | --- |
| Имя | Разрядность | Описание |
| alu\_resul\_mem(or address for dl1) | [31:0] | Результат вычисления ALU |
| sx\_imm\_mem | [31:0] | Операнд с кода команды |
| hz\_bus\_mem | [16:0] | Шина передачи данных hazard\_ctrl(описание в таблице выше) |
| pc\_4 | [31:0] | Текущее значение программного счетчика + 4 |
| we\_reg\_file\_dec | [0:0] | Разрешение записи в регистровый файл |

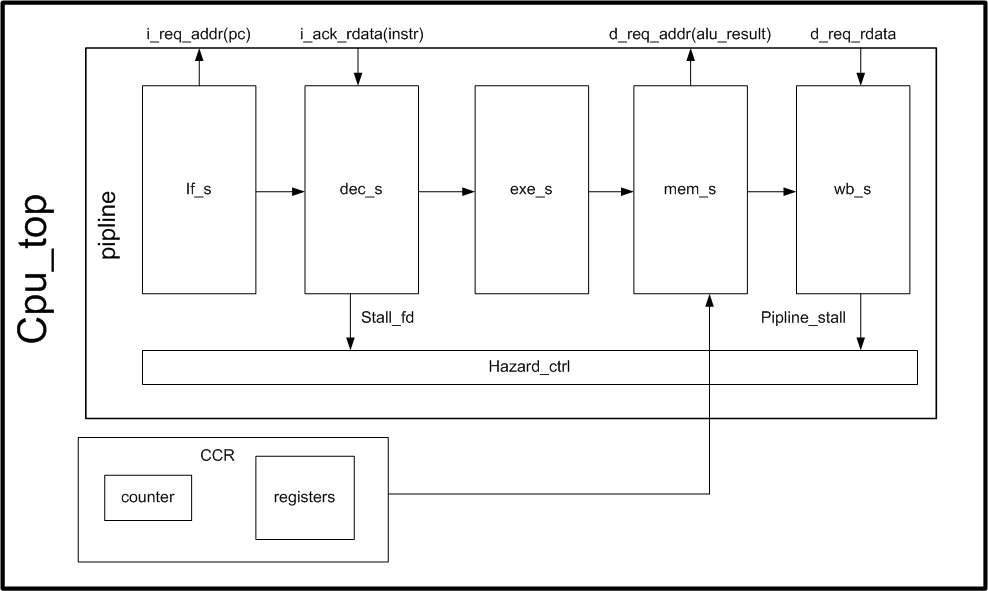


Рис 2 Блок схема модулей и их взаимодействия в процессорном ядре

## Микроархитектура

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

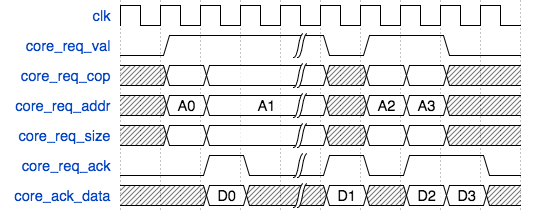


Рисунок 1

Ядро выставляет core\_req\_val для подтверждения валидности данных на шине и вместе с этим код операции core\_req\_cop адрес обращения core\_req\_addr и размер запроса core\_req\_size. Со стороны кэша выставляется core\_req\_ack для подтверждения валидности данных и вместе с этим данные core\_ack\_data.

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

## 

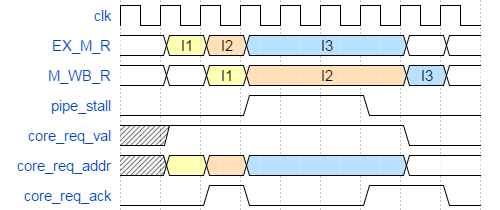
**Таблица 1. Интерфейс взаимодействия ядра с кэш-памятью инструкций**

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



При обращение в память ядро сообщает подсистеме памяти какой тип запроса: кэшируемой обращение или некэшируемой. В управляющих регистрах ядра хранится информация об окне некэширемых адресов

**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\_req\_ack** | **[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